CS336 project 1

part 1:

1. 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.

```
DELIMITER $$
create procedure API1(in C varchar(50), in T varchar(50), in P varchar(50))
Begin
        select (case
   when C not in('Trump', 'Biden') then
   'incorrect candidate'
   when P not in(select distinct precinct from penna) then
   'incorrect precinct'
   when T not like '%-%-% %:%:%'
   or substring_index(T, '-', 1) < 2020
   or substring_index(substring_index(T, '-', 2), '-', -1) > 12
   or substring_index(substring_index(T, ' ', 1), '-', -1) > 31
   or substring_index(substring_index(T, ':', 1), ' ', -1) > 24
   or substring_index(substring_index(T, ':', 2), ':', -1) > 59
   or substring_index(T, ':', -1) > 59 then 'incorrect timestamp'
   when C = 'Trump' then
        (select Trump from penna where Timestamp = (select Timestamp from penna where
Timestamp <= T order by Timestamp desc limit 1) AND precinct = P)
   when C = 'Biden' then
   (select Biden from penna where Timestamp = (select Timestamp from penna where
Timestamp <= T order by Timestamp desc limit 1) AND precinct = P)
        end) as result;
end $$
DELIMITER ;
```

2. 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.

```
DELIMITER $$
create procedure API2(in D varchar(50))
Begin
        if D not like '%-%-%' or substring_index(D, '-', 1) < 2020 or
substring_index(substring_index(D, '-', 2), '-', -1) > 12 or substring_index(D, '-', -1) > 31
   then select 'incorrect timestamp' as API2;
   else
```

```
          select (case
    when a.sumT < a.sumB then concat('Biden ', a.sumB) when a.sumT > a.sumB then
concat('Trump ', a.sumT) else 'tie'
    end) as result from
    (select distinct timestamp, sum(Biden) as sumB, sum(Trump) as sumT from penna where
timestamp =
    (select distinct timestamp from penna where timestamp like concat(D, '%') order by
timestamp desc limit 1) group by timestamp) a;
    end if;
end $$
DELIMITER ;
```

3. API3(candidate) - Given a candidate return top 10 precincts that this candidate win. Order precincts by total votes and list TOP 10 in descending order of totalvotes.

```
DELIMITER $$
create procedure API3(in C varchar(50))
Begin
          if C = 'Trump' then
          select distinct precinct from penna where Trump > Biden AND timestamp = (select
max(timestamp) from penna) order by totalvotes desc limit 10;
    else if C = 'Biden' then
    select distinct precinct from penna where Biden > Trump AND timestamp = (select
max(timestamp) from penna) order by totalvotes desc limit 10;
    else if C not in('Trump', 'Biden') then
    select 'incorrect candidate';
    end if;
    end if;
    end if;
end $$
DELIMITER ;
```

4. 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.

```
DELIMITER $$
create procedure API4(in P varchar(50))
begin
if P not in(select distinct precinct from penna) then
 select 'incorrect precinct' as API4;
 else
 select (case
 when a.sumT > a.sumB then
 concat('Trump ',concat((a.sumT/a.total)*100,"%"))
 when a.sumT < a.sumB then
 concat('Biden ', concat((a.sumB/a.total)*100,"%"))
 else 'tie'
 end) as result from
 (select distinct sum(totalvotes) as total, sum(Biden) as sumB, sum(Trump) as sumT
 from penna where precinct = (select distinct precinct from penna where precinct = P) group by
precinct) a;
 end if;
end $$
DELIMITER ;
```

'No sum version'

```
DELIMITER $$
create procedure API4_noSum (in P varchar(50))
begin
if P not in(select distinct precinct from penna) then
 select 'incorrect precinct' as API4;
 else
 select (case
 when a.Trump > a.Biden then
 concat('Trump ',concat((a.Trump/a.totalvotes)*100,"%"))
 when a.Trump < a.Biden then
 concat('Biden ', concat((a.Biden/a.totalvotes)*100,"%"))
 else 'tie'
 end) as result from
 (select distinct totalvotes, Biden, Trump
 from penna where timestamp = (select max(timestamp) from penna where precinct = P)
 AND precinct = (select distinct precinct from penna where precinct = P)) a;
 end if;
end $$
DELIMITER ;
```

5. 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 precincts which have "Township" in their name as well as sum of votes for the winner.

```
DELIMITER $$
create procedure API5(in S varchar(50))
begin
	if not exists(select precinct from penna where precinct like concat('%', S, '%')) then select
'incorrect string' as API5;
```

```
   else
         select (case when a.trump > a.biden then concat('Trump win: ', a.Trump) when a.trump <
a.biden then concat('Biden win : ', a.Biden) else 'tie' end) as result from
   (select sum(Trump) as trump, sum(Biden) as biden from penna where precinct like
concat('%', S, '%') AND timestamp = (select max(timestamp) from penna where precinct like
concat('%', S, '%'))) a;
   end if;
end $$
DELIMITER ;
```

part 2:

1) newPenna(): 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 last 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.

```
DELIMITER $$
create procedure newPenna()
begin
        CREATE TABLE newPenna
        select a.precinct, a.timestamp, a.totalvotes - b.nextTotal as add_Total, a.Biden -
b.nextBiden as add_Biden, a.Trump - b.nextTrump as add_Trump from
        (select precinct, totalvotes, Biden, Trump, timestamp from Penna) a,
        (select precinct, totalvotes, Biden, Trump, timestamp,
        Lead(totalvotes, 1, 0) OVER(partition by precinct ORDER BY precinct, timestamp desc)
AS nextTotal,
        lead(Biden, 1, 0) OVER(partition by precinct ORDER BY precinct, timestamp desc) AS
nextBiden,
        lead(Trump, 1, 0) OVER(partition by precinct ORDER BY precinct, timestamp desc) AS
nextTrump FROM penna) b
        where a.precinct = b.precinct AND a.timestamp = b.timestamp Order by a.precinct,
a.Timestamp desc;
end $$
DELIMITER ;
```

2) Switch(): 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.   The format of the table should be:

**Switch(precinct, timestamp, fromCandidate, toCandidate) where fromCandidate is the candidate who was leading at timestamp in precinct,
but he lost the lead to the toCandidate (who maintained that lead till the end)**

```
DELIMITER $$
create procedure Switch()
begin
        create table Switch
        select distinct
        if(d.winner1 != c.winner, d.precinct, '') as precincts,
        if(d.winner1 != c.winner, d.timestamp, '') as timestamps,
        if(d.winner1 != c.winner, d.winner1, '') as fromCandidate,
        if(d.winner1 != c.winner, c.winner, '') as toCandidate from
        (select distinct p.precinct, p.timestamp, totalvotes, Trump, Biden, (case when p.Trump -
p.Biden > 0 then 'Trump' when p.Trump - p.Biden < 0 then 'Biden' end) as winner from
        (select distinct precinct, max(timestamp) Over(partition by precinct) as maxTime from
penna) a, penna p
        where p.precinct = a.precinct AND p.timestamp = a.maxTime) c,
        (select distinct p.precinct, p.timestamp, totalvotes, Trump, Biden, (case when p.Trump -
p.Biden > 0 then 'Trump' when p.Trump - p.Biden < 0 then 'Biden' end) as winner1 from
        (select distinct precinct, max(timestamp) Over(partition by precinct) as maxTime from
penna) a, penna p
        where p.timestamp >= subtime(a.maxTime, '24:00:00') AND p.precinct = a.precinct order
by precinct, timestamp desc) d
        where c.precinct = d.precinct;
end $$
DELIMITER ;
```

part3

a) The sum of votes for Trump and Biden cannot be larger than totalvotes

```
select (case
   when exists(select * from penna where (trump + biden) > totalvotes)
   then 'FALSE'
   else 'TRUE'
   end) as result;
```

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

```
select (case
        when a.min < '2020-11-03' or a.max > '2020-11-11'
   then 'FALSE'
```

```
    else 'TRUE'
end) as result from (select substring_index(min(timestamp), ' ', 1) as min,
substring_index(max(timestamp), ' ', 1) as max from penna) a
```

c) Totalvotes for any precinct and at any timestamp T > 2020-11-05 00:00:00, will be smaller than totalvotes at T'<T but T'>2020-11-05 00:00:00 for that precinct.

'timestamp greater or equal to 2020-11-05'

```
select distinct if(count(*) > 1, 'FALSE', 'TRUE') as result from
(select distinct if (c.add_Total < 0, 'FALSE', 'TRUE') as TF from
(select a.precinct, a.timestamp, a.totalvotes - b.nextTotal as add_Total, a.Biden - b.nextBiden as
add_Biden, a.Trump - b.nextTrump as add_Trump from
(select precinct, totalvotes, Biden, Trump, timestamp from Penna) a,
(select precinct, totalvotes, Biden, Trump, timestamp,
Lead(totalvotes, 1, 0) OVER(partition by precinct ORDER BY precinct, timestamp desc) AS
nextTotal,
lead(Biden, 1, 0) OVER(partition by precinct ORDER BY precinct, timestamp desc) AS
nextBiden,
lead(Trump, 1, 0) OVER(partition by precinct ORDER BY precinct, timestamp desc) AS
nextTrump FROM penna) b
where a.precinct = b.precinct AND a.timestamp = b.timestamp AND a.timestamp > '2020-11-05
00:00:00' Order by a.precinct, a.Timestamp desc) c) d;
```

'timestamp only in 2020-11-05'

```
select distinct if(count(*) > 1, 'FALSE', 'TRUE') as result from
(select distinct if (c.add_Total < 0, 'FALSE', 'TRUE') as TF from
(select a.precinct, a.timestamp, a.totalvotes - b.nextTotal as add_Total, a.Biden - b.nextBiden as
add_Biden, a.Trump - b.nextTrump as add_Trump from
(select precinct, totalvotes, Biden, Trump, timestamp from Penna) a,
(select precinct, totalvotes, Biden, Trump, timestamp,
Lead(totalvotes, 1, 0) OVER(partition by precinct ORDER BY precinct, timestamp desc) AS
nextTotal,
lead(Biden, 1, 0) OVER(partition by precinct ORDER BY precinct, timestamp desc) AS
nextBiden,
lead(Trump, 1, 0) OVER(partition by precinct ORDER BY precinct, timestamp desc) AS
nextTrump FROM penna) b
where a.precinct = b.precinct AND a.timestamp = b.timestamp AND a.timestamp like
'%2020-11-05%' Order by a.precinct, a.Timestamp desc) c) d;
```

part 4

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. Initially theses tables should be empty. In your video you should show how updates, insertions and deletions result in adding tuples to the Updated Tuples, Inserted Tuples and Deleted Tuples. Show multiple examples of deletions, insertions and updates, and display these tables after each of them.

```
CREATE TABLE `InsertTuples` (
`ID` int(11) DEFAULT NULL,
`Timestamp` datetime DEFAULT NULL,
`state` varchar(10) DEFAULT NULL,
`locality` varchar(100) DEFAULT NULL,
`precinct` varchar(100) DEFAULT NULL,
`geo` varchar(100) DEFAULT NULL,
`totalvotes` int(11) DEFAULT NULL,
`Biden` int(11) DEFAULT NULL,
`Trump` int(11) DEFAULT NULL,
`filestamp` varchar(200) DEFAULT NULL,
KEY `indp` (`precinct`),
KEY `indtemp` (`Timestamp`),
KEY `indprecon` (`precinct`)
);

CREATE TABLE `DeletedTuple` (
`ID` int(11) DEFAULT NULL,
`Timestamp` datetime DEFAULT NULL,
`state` varchar(10) DEFAULT NULL,
`locality` varchar(100) DEFAULT NULL,
`precinct` varchar(100) DEFAULT NULL,
`geo` varchar(100) DEFAULT NULL,
`totalvotes` int(11) DEFAULT NULL,
`Biden` int(11) DEFAULT NULL,
`Trump` int(11) DEFAULT NULL,
`filestamp` varchar(200) DEFAULT NULL,
KEY `indp` (`precinct`),
KEY `indtemp` (`Timestamp`),
KEY `indprecon` (`precinct`)
);

CREATE TABLE `UpdatedTuple` (
`ID` int(11) DEFAULT NULL,
`Timestamp` datetime DEFAULT NULL,
`state` varchar(10) DEFAULT NULL,
`locality` varchar(100) DEFAULT NULL,
`precinct` varchar(100) DEFAULT NULL,
`geo` varchar(100) DEFAULT NULL,
`totalvotes` int(11) DEFAULT NULL,
`Biden` int(11) DEFAULT NULL,
`Trump` int(11) DEFAULT NULL,
`filestamp` varchar(200) DEFAULT NULL,
KEY `indp` (`precinct`),
KEY `indtemp` (`Timestamp`),
KEY `indprecon` (`precinct`)
);

select * from InsertTuples;

delimiter $$
create trigger insert_partFour
before insert on Penna
for each row
begin
insert into InsertTuples value (new.ID, NEW.Timestamp, NEW.state, NEW.locality,
NEW.precinct, NEW.geo, NEW.totalvotes, NEW.Biden, NEW.Trump, NEW.filestamp);
end $$
delimiter ;

select * from penna where precinct = 'aaaaaaaa';
insert into Penna values (111111, "2020-12-12 23:15:45", "PA", "C", "aaaaaaaa",
"42021-ADAMS TWP DUNLO", 0, 0, 0, "NOVEMBER_04_2020_035836.json");
select * from InsertTuples;

delimiter $$
create trigger delete_partFour
before delete on Penna
for each row
begin
insert into DeletedTuple value (old.ID, old.Timestamp, old.state, old.locality, old.precinct,
old.geo, old.totalvotes, old.Biden, old.Trump, old.filestamp);
end $$
delimiter ;

select * from penna where precinct = 'aaaaaaaa';
delete From Penna where precinct = "aaaaaaaa";
select * from DeletedTuple;
```

```
delimiter $$
create trigger update_partFour
before update on Penna
for each row
begin
insert into UpdatedTuple value (old.ID, old.Timestamp, old.state, old.locality, old.precinct,
old.geo, old.totalvotes, old.Biden, old.Trump, old.filestamp);
end $$
delimiter ;

select * from penna where precinct = 'aaaaaaaa';
update penna set precinct = 'aaaaaaaa' where precinct = 'bbbbbbb';
select * from UpdatedTuple;
```

4.2)

MoveVotes(Precinct, Timestamp, Candidate, Number_of_Moved_Votes)
a) Precinct – one of the existing precincts
b) Timestamp must be existing timestamp. If Timestamp does not appear in Penna than MoveVotes should display a message "Unknown Timestamp".
c) 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".
d) Of course if CoreCandidate is neither Trump nor Biden, MoveVotes() should say "Wrong Candidate".

"update version of MoveVotes (will update the original table)"

```
SET SQL_SAFE_UPDATES = 0;
DELIMITER $$
create procedure MoveVotes(in P varchar(50), in T varchar(50), in Candidate varchar(50), in
Number_of_Moved_Votes varchar(50))
begin
        if Candidate not in('Trump', 'Biden') then
        select 'Wrong Candidate' as MoveVotes;
        else if P not in(select distinct precinct from penna) then
   select 'incorrect precinct' as MoveVotes;
   else if T not in(select distinct timestamp from penna) then
   select 'Unknown Timestamp' as MoveVotes;
   else if Candidate = 'Biden' AND Number_of_Moved_Votes > (select Biden from penna where
timestamp = T AND precinct = P) or
   Candidate = 'Trump' AND Number_of_Moved_Votes > (select Trump from penna where
timestamp = T AND precinct = P) then
   select 'Not enough votes' as MoveVotes;
```

```
        else if Candidate = 'Biden' AND Number_of_Moved_Votes <= (select Biden from penna
where timestamp = T AND precinct = P) then
        update penna
   set Trump = Trump + Number_of_Moved_Votes, Biden = Biden - Number_of_Moved_Votes
   where precinct = P AND timestamp >= T;
   else if Candidate = 'Trump' AND Number_of_Moved_Votes <= (select Trump from penna
where timestamp = T AND precinct = P) then
        update penna
   set Trump = Trump - Number_of_Moved_Votes, Biden = Biden + Number_of_Moved_Votes
   where precinct = P AND timestamp >= T;
   end if;
   end if;
   end if;
   end if;
   end if;
   end if;
end $$
DELIMITER ;
```

"select version of MoveVotes (will not update the original table)"

```
DELIMITER $$
create procedure MoveVotes2(in P varchar(50), in T varchar(50), in Candidate varchar(50), in
Number_of_Moved_Votes varchar(50))
begin
        if Candidate not in('Trump', 'Biden') then
        select 'Wrong Candidate' as MoveVotes;
        else if P not in(select distinct precinct from penna) then
   select 'incorrect precinct' as MoveVotes;
   else if T not in(select distinct timestamp from penna) then
   select 'Unknown Timestamp' as MoveVotes;
   else if Candidate = 'Biden' AND Number_of_Moved_Votes > (select Biden from penna where
timestamp = T AND precinct = P) or
   Candidate = 'Trump' AND Number_of_Moved_Votes > (select Trump from penna where
timestamp = T AND precinct = P) then
   select 'Not enough votes' as MoveVotes;
        else if Candidate = 'Biden' AND Number_of_Moved_Votes <= (select Biden from penna
where timestamp = T AND precinct = P) then
   select distinct precinct, timestamp, totalvotes, (Biden - Number_of_Moved_Votes) as
newBiden, (Trump + Number_of_Moved_Votes) as newTrump from penna
   where precinct = P AND timestamp >= T order by timestamp desc;
        else if Candidate = 'Trump' AND Number_of_Moved_Votes <= (select Trump from penna
where timestamp = T AND precinct = P) then
        select distinct precinct, timestamp, totalvotes, (Biden + Number_of_Moved_Votes) as
newBiden, (Trump - Number_of_Moved_Votes) as newTrump from penna
```

```
    where precinct = P AND timestamp >= T order by timestamp desc;
    end if;
    end if;
    end if;
    end if;
    end if;
    end if;
end $$
DELIMITER ;
```