# I am building a database for federal elections in Australia. Here are my tables:

CREATE TABLE TESTelectionMaster (

    electionSerialNo INTEGER PRIMARY KEY,

    electionDate date,

    electionType VARCHAR(50),

    totalNumDivisions INTEGER,

    totalRegVoters INTEGER,

    lastDateToVoterRegister DATE

    lastDateCandidateNominate DATE,

    lastDateToDeclareResult DATE

);

CREATE TABLE TESTelectoralDivision (

    divisionName VARCHAR(50) PRIMARY KEY,

    totalRegVoters INTEGER,

    currMember VARCHAR(50),

);

CREATE TABLE TESTelectoralDivisionHistory (

    divisionName VARCHAR(50),

    electionSerialNo Integer,

    historicRegVoters INTEGER,

    PRIMARY KEY (divisionName, electionSerialNo)

    );

CREATE TABLE TESTelectionEvent (

    electionEventID VARCHAR,

    totalVoters INTEGER,

    votesCast INTEGER,

    votesReject INTEGER,

    votesValid INTEGER,

    electionSerialNo INTEGER,

    divisionName VARCHAR(50),

    prefWinnerCandidateID VARCHAR(20),

    winnerTally Integer,

    prefLoserCandidateID VARCHAR(20),

    loserTally INTEGER,

    PRIMARY KEY (electionEventID)

);

CREATE TABLE TESTcandidateList (

    candidateID VARCHAR(20),

    candidateName VARCHAR (50),

    contactAddress VARCHAR (50),

    contactPhone INTEGER,

    contactMobile INTEGER,

    contactEmail VARCHAR(20),

    partyCode VARCHAR(60),

    PRIMARY KEY (candidateID)

);

CREATE TABLE TESTcontests (

    electionEventID VARCHAR,

    candidateID VARCHAR (20),

    PRIMARY KEY (electionEventID, candidateID)

);

CREATE TABLE TESTpoliticalParty (

    partyCode VARCHAR(60) PRIMARY KEY,

    partyName VARCHAR(100),

    partyLogo TEXT,

    postalAddress VARCHAR(50),

    partySecretary VARCHAR(50),

    contactPersonName VARCHAR(50),

    contactPersonPhone VARCHAR(20),

    contactPersonMobile VARCHAR(20),

    contactPersonEmail VARCHAR(50)

);

CREATE TABLE TESTvoterRegistry (

    title VARCHAR(5),

    voterID INTEGER,

    firstName VARCHAR(60),

    middleName VARCHAR(60),

    lastName VARCHAR(60),

    gender VARCHAR(60),

    dateOfBirth DATE,

    residentUnitNumber VARCHAR(20),

    residentStreetNumber INTEGER,

    residentStreetName VARCHAR(60),

    residentsuburb VARCHAR(30),

    residentPostcode INTEGER,

    residentState VARCHAR(30),

    postalUnitNumber INTEGER,

    postalStreetNumber INTEGER,

    postalStreetName VARCHAR(60),

    postalSuburb VARCHAR(60),

    postalPostcode INTEGER,

    postalState VARCHAR(60),

    daytimePhone INTEGER,

    mobile VARCHAR(20),

    emailAddress VARCHAR(40),

    divisionName VARCHAR(50),

    PRIMARY KEY (voterID)

);

CREATE TABLE TESTballot (

    ballotID INTEGER,

    electionEventID VARCHAR,

    PRIMARY KEY (ballotID)

);

CREATE TABLE TESTballotPreferences (

    ballotID INTEGER,

    candidateID VARCHAR (20),

    preference INTEGER,

    PRIMARY KEY (ballotID, candidateID)

);

CREATE TABLE TESTissuanceRecord (

    voterID INTEGER,

    electionEventID VARCHAR,

    issueDate DATE,

    ballotIssue Timestamp,

    pollingStation VARCHAR(50),

    PRIMARY KEY (voterID, electionEventID)

);

CREATE TABLE TESTprefCountRecord (

    electionEventID VARCHAR,

    roundNo INTEGER,

    eliminatedCandidateID VARCHAR (20),

    countStatus VARCHAR, --Done, In-progress, complete

    preferenceAggregate INTEGER,

    PRIMARY KEY (electionEventID, roundNo)

);

CREATE TABLE TESTpreferenceTallyPerRoundPerCandidate (

    electionEventID VARCHAR,

    roundNo integer,

    candidateID VARCHAR(20),

    preferenceTally INTEGER, -- Tally in a round.

    PRIMARY KEY (electionEventID, roundNo, candidateID)

);

Here are there constraints:

alter table TESTelectoralDivisionHistory add constraint TESTelectoralDivisionHistoryKeys

    FOREIGN KEY (divisionName)

        REFERENCES TESTelectoralDivision (divisionName),

    FOREIGN KEY (electionSerialNo)

        REFERENCES TESTelectionMaster (electionSerialNo);

alter table TESTcandidateList add constraint TESTcandidateListKeys

    FOREIGN KEY (partyCode)

        REFERENCES TESTpoliticalParty (partyCode);

alter table TESTvoterRegistry add constraint TESTvoterRegistryKeys

    FOREIGN KEY (divisionName)

        REFERENCES TESTelectoralDivision (divisionName);

alter table TESTcontests add constraint TESTcontestsKeys

    FOREIGN KEY (electionEventID)

        REFERENCES TESTelectionEvent (electionEventID),

    FOREIGN KEY (candidateID)

        REFERENCES TESTcandidateList (candidateID);

alter table TESTballot add constraint TESTballotKeys

    FOREIGN KEY (electionEventID)

        REFERENCES TESTelectionEvent (electionEventID);

alter table TESTballotPreferences add constraint TESTballotPreferencesKeys

    FOREIGN KEY (ballotID)

        REFERENCES TESTballot(ballotID),

    FOREIGN KEY (candidateID)

        REFERENCES TESTcandidateList (candidateID);

alter table TESTissuanceRecord add constraint TESTissuanceRecordKeys

    FOREIGN KEY (voterID)

        REFERENCES TESTvoterRegistry (voterID),

    FOREIGN KEY (electionEventID)

        REFERENCES TESTelectionEvent (electionEventID);

alter table TESTprefCountRecord add constraint TESTprefCountRecordKeys

    FOREIGN KEY (electionEventID)

        REFERENCES TESTelectionEvent (electionEventID),

    FOREIGN KEY (eliminatedCandidateID)

        REFERENCES TESTcandidateList (candidateID);

alter table TESTpreferenceTallyPerRoundPerCandidate add constraint TESTpreferenceTallyPerRoundPerCandidateKeys

    FOREIGN KEY (electionEventID)

        REFERENCES TESTelectionEvent (electionEventID),

    FOREIGN KEY (electionEventID, roundNo)

        REFERENCES TESTprefCountRecord (electionEventID, roundNo),

    FOREIGN KEY (candidateID)

        REFERENCES TESTcandidateList (candidateID);

alter table TESTelectionEvent add constraint TESTelectionEventKeys

    FOREIGN KEY (electionSerialNo)

        REFERENCES TESTelectionMaster (electionSerialNo),

    FOREIGN KEY (divisionName)

        REFERENCES TESTelectoralDivision (divisionName),

    FOREIGN KEY (prefWinnerCandidateID)

        REFERENCES TESTcandidateList (candidateID),

    FOREIGN KEY (prefLoserCandidateID)

        REFERENCES TESTcandidateList (candidateID);

**Here are my indexes:**

CREATE INDEX idx\_voterRegistry\_divisionName ON TESTvoterRegistry(divisionName);

CREATE INDEX idx\_election\_ballot\_pref\_candidate ON

    TESTelectionEvent (electionEventID),

    TESTballot (electionEventID),

    TESTballotPreference (ballotID, candidateID),

    TESTcandidateList (candidateID, partyCode),

    TESTpoliticalParty (partyCode);

CREATE INDEX idx\_election\_ballot\_pref\_candidate ON

    TESTelectionEvent (electionEventID),

    TESTballot (electionEventID),

    TESTballotPreference (ballotID, candidateID),

    TESTcandidateList (candidateID, partyCode),

    TESTpoliticalParty (partyCode);

Can you create a partitioning strategy for TESTvoterRegistry, TESTballot & TESTballotPreferences? Answer yes or no.

# Create a partioning strategy for TESTvoterRegistry

## Here are queries and indexes related to TESTvoterRegistry, TESTballot & TESTballotPreferences:

SELECT divisionName, COUNT(voterID) AS totalNumVoters

FROM TESTvoterRegistry

GROUP BY divisionName

ORDER BY totalNumVoters DESC;

SELECT

    e.divisionName,

    c.candidateName,

    p.partyName

FROM

    TESTelectionEvent e

JOIN

    TESTballot b ON e.electionEventID = b.electionEventID

JOIN

    TESTballotPreference bp ON b.ballotID = bp.ballotID

JOIN

    TESTcandidateList c ON bp.candidateID = c.candidateID

JOIN

    TESTpoliticalParty p ON c.partyCode = p.partyCode

WHERE

    e.electionEventID = '20220521' -- Format from 2019 assignment specs

ORDER BY

    e.divisionName,

    RANDOM();

SELECT

    v.firstName,

    v.middleName,

    v.lastName,

    v.residentUnitNumber,

    v.residentStreetNumber,

    v.residentStreetName,

    v.residentsuburb,

    v.residentPostcode,

    v.residentState

FROM

    TESTvoterRegistry v

WHERE

    v.voterID NOT IN (

        SELECT b.voterID

        FROM TESTballot b

        WHERE b.electionEventID IN ('20220521', '20190518')

    )

ORDER BY

    v.lastName, v.firstName;

Since I am programming in SQL SERVER, ONLY range partionings can be accepted. Just to check if I programmed in another SQL language (E.g. PostSQL) for TESTvoterRegistry would it work to partition divisionName and voterID to do a partion-wise join?

**Create**

1. Define the partitioning strategy – groups.

* This must be programmed in SQLSERVER which only accepts Range partitioning methods.
* Include details of the partitioning type and which columns/partition key to be used.
* You must include the SQL DDL statements used to implement your partition strategy.

You must justify your design decisions. Include details about which of the above queries it will improve the performance of and how it helps with concrete examples. (You must explain in clear terms – such as partition pruning, partition joins, and parallel SQL, applicable to each of these queries.)

1. Define the partitioning scheme – how individual partitions are stored.

* Include details of the partitioning type and which columns/partition key to be used.
* You must include the SQL DDL statements used to implement your partition strategy.

You must justify your design decisions. Include details about which of the above queries it will improve the performance of and how it helps with concrete examples. (You must explain in clear terms – such as partition pruning, partition joins, and parallel SQL, applicable to each of these queries.)