The Knicks' 416 Database Schema

*****ALL FIELDS ARE SUBJECT TO CHANGE. ALL CHANGES WILL BE DOCUMENTED IN THE "sql" FOLDER OF THE PROJECT AND WILL BE REFLECTED IN FUTURE SCHEMA PDF ADAPTATIONS*****

Table: app.states

Purpose: Stores state-level information including mapping, registration policy flags, and population stats.

Name	Type	Details
state_id	SERIAL PRIMARY KEY	Unique identifier for each state.
name	VARCHAR(50) NOT NULL	Full state name.
code	CHAR(2) NOT NULL UNIQUE	Two-letter USPS code; unique constraint prevents duplicates.
geom_boundary	TEXT	Boundary polygon serialized as text (no PostGIS).
geom_center	TEXT	Visual centroid (lon,lat) stored as text.
map_zoom_level	INT NOT NULL	Recommended default zoom for map UI.
registration_method	VARYING(7)	Registration policy (e.g., 'opt-in','opt-out'); updated per request.
same_day_registration	BOOLEAN	True if SDR is allowed.
felony_disenfranchisement	SMALLINT	Policy scale 1–4 (documentation-defined).
population_total	INT	Total population.
citizens_of_voting_age_population	INT	Eligible citizens by age (CVAP).
house_seats_rep	INT	Number of Republican U.S. House seats.
house_seats_dem	INT	Number of Democratic U.S. House seats.
redistricting_control	VARCHAR(20)	Controller (party/commission).
dominant_party	CHAR(1)	Dominant party ('R' or 'D').

Indexes (what they speed up)

- PRIMARY KEY (state_id): unique row lookups and FK joins to states. - UNIQUE (code): prevents duplicate USPS codes. - idx_states_name: faster case-insensitive searches by name (use LOWER(name)). - idx_states_redistricting: quick filters on redistricting_control for analytics.

Table: app.eavs_facts

Purpose: Stores EAVS quantitative facts by region and year.

Name	Type	Details
region_id	VARCHAR(10) NOT NULL	Jurisdiction identifier (string code).
year	INT NOT NULL	Election year.
active_registered	INT	Active registered voters.
inactive_registered	INT	Inactive registered voters.
total_registered	INT	Total registered voters.
reg_missing_data	BOOLEAN	True if registration section is missing/unknown.
total_removed	INT	Total removals.
removed_moved	INT	Removals due to moving.
removed_felony	INT	Removals due to felony.
removed_deceased	INT	Removals due to death.
removed_failed_confirm	INT	Removals due to failed confirmation.
removed_incompetent	INT	Removals due to incompetence.
removed_requested	INT	Removals by voter request.
removed_duplicate	INT	Removals due to duplicates.
removed_other	INT	Other removal reasons.
del_missing_data	BOOLEAN	True if deletion data missing.
total_ballots_cast	INT	Total ballots cast.
ballots_by_mail	INT	Mail ballots cast.
ballots_in_person_early	INT	Early in-person ballots.
ballots_in_person_eday	INT	Election Day in-person ballots.
ballots_dropbox	INT	Dropbox ballots.
early_voting_total	INT	Total early votes.
early_missing_data	BOOLEAN	True if early voting data missing.
prov_cast	INT	Provisional ballots cast.
prov_reason_not_in_roll	INT	Provisional reason: not in roll.
prov_reason_no_id	INT	Provisional reason: no ID.
prov_reason_not_eligible_official	INT	Provisional reason: eligibility issue.
prov_reason_challenged	INT	Provisional reason: challenged.
prov_reason_wrong_precinct	INT	Provisional reason: wrong precinct.
prov_reason_name_address	INT	Provisional reason: name/address issue.
prov_reason_mail_ballot_unsurrendered	INT	Provisional reason: mail ballot not surrendered.
prov_reason_hours_extended	INT	Provisional reason: extended hours.
prov_reason_same_day_reg	INT	Provisional reason: same-day registration.
prov_other	INT	Other provisional reasons.
mail_reject_total	INT	Rejected mail ballots: total.
mail_reject_late	INT	Rejected: late.
mail_reject_no_sig	INT	Rejected: missing signature.
mail_reject_no_witness_sig	INT	Rejected: missing witness signature.
mail_reject_sig_mismatch	INT	Rejected: signature mismatch.
mail_reject_unofficial_env	INT	Rejected: unofficial envelope.
mail_reject_ballot_missing	INT	Rejected: missing ballot.

mail_reject_no_secrecy_env	INT	Rejected: no secrecy envelope.
mail_reject_multiple_in_env	INT	Rejected: multiple ballots in envelope.
mail_reject_unsealed_env	INT	Rejected: unsealed envelope.
mail_reject_no_postmark	INT	Rejected: no postmark.
mail_reject_no_address	INT	Rejected: missing address.
mail_reject_voter_deceased	INT	Rejected: voter deceased.
mail_reject_duplicate_vote	INT	Rejected: duplicate vote.
mail_reject_missing_docs	INT	Rejected: missing documents.
mail_reject_not_eligible	INT	Rejected: voter not eligible.
mail_reject_no_application	INT	Rejected: no application.
mail_reject_other	INT	Rejected: other reasons.
mail_missing_data	BOOLEAN	True if mail data missing.
missing_data_score	NUMERIC	Score reflecting data completeness.

Primary Key & Indexes (what they speed up)

- PRIMARY KEY (region_id, year): direct point lookup by region-year. - idx_eavs_facts_year: fast filtering and grouping by year. - idx_eavs_facts_region: fast time-series queries for a single region.

Table: app.census_block

Purpose: Census block centroids for selected states (text geometry).

Name	Type	Details
block_id	CHAR(15) PRIMARY KEY	Census block FIPS identifier.
state_id	INT NOT NULL REFERENCES app.sates(state_id)	
geom_center	TEXT NOT NULL	Centroid serialized as text.

Indexes (what they speed up)

- PRIMARY KEY (block_id): direct lookups and joins from voter registration. - idx_census_block_state_id: state-level filtering of blocks.

Table: app.device_model

Purpose: Catalog of voting devices, vendors, and key attributes.

Name	Type	Details
device_model_id	SERIAL PRIMARY KEY	Unique device model identifier.
vendor	VARCHAR(50) NOT NULL	Manufacturer name.
model_name	VARCHAR(50) NOT NULL	Model designation.
device_type	VARCHAR(20) NOT NULL	Category, e.g., 'scanner','BMD'.
description	TEXT	Freeform description.
year_introduced	SMALLINT	First release year.
certification	VARCHAR(20)	Certification standard.
underlying_os	VARCHAR(30)	Operating system family.
scan_rate	SMALLINT	Approx. pages/minute or ballots/minute.
error_rate	DECIMAL(4,3)	Measured error rate (0–1.000).
reliability	DECIMAL(3,1)	Reliability score (0–10).
quality_score	DECIMAL(3,2)	Composite quality metric.
is_discontinued	BOOLEAN DEFAULT FALSE	True if model is discontinued.

Indexes (what they speed up)

- UNIQUE (vendor, model_name): prevents duplicate entries for the same model. - idx_device_model_vendor: vendor-based filters. - idx_device_model_type: device-type category screens. - idx_device_model_certification: compliance-oriented queries. - idx_device_model_year: reports grouped by year introduced.

Table: app.equipment_usage

Purpose: Device deployment counts per region and year.

Name	Type	Details
usage_id	SERIAL PRIMARY KEY	Unique record ID.
state_id	INT NOT NULL REFERENCES app.states(state_id)	
region_id	VARCHAR(10)	Region/jurisdiction code (string).
year	INT NOT NULL	Election year.
device_model_id	INT NOT NULL REFERENCES app.device_model(device_mode_id)	.
quantity	INT NOT NULL DEFAULT 0	Number of devices deployed.
avg_age	DECIMAL(4,1)	Average age of devices in years.

Indexes (what they speed up)

- idx_equipment_usage_state: state-level filtering. - idx_equipment_usage_region: per-region drilldowns. - idx_equipment_usage_year: dashboards by year. - idx_equipment_usage_device: device-centric searches. - idx_equipment_usage_state_year: common state/year dashboards. - idx_equipment_usage_region_year: region/year charts and comparisons.

Table: app.voter_registration

Purpose: Registered voter records for detailed states (PII minimized).

Name	Type	Details
voter_id	SERIAL PRIMARY KEY	Unique voter identifier.
state_id	INT NOT NULL REFERENCES app.states(state_id)	.
region_id	VARCHAR(10)	Region/jurisdiction code (string).
first_name	VARCHAR(50)	First name.
last_name	VARCHAR(50)	Last name.
middle_name	VARCHAR(50)	Middle name.
party_affiliation	VARCHAR(20)	Party affiliation text.
status	VARCHAR(10)	Voter status.
city	VARCHAR(50)	City of residence.
zip_code	VARCHAR(10)	Postal ZIP code.
residential_address	VARCHAR(120)	Street address (normalized as needed).
registration_date	DATE	Date of registration.
census_block_id	CHAR(15) REFERENCES app.census_block(block_id)	

Indexes (what they speed up)

- idx_voter_registration_state: filter by state. - idx_voter_registration_region: jurisdiction filters. - idx_voter_registration_party: party-based queries. - idx_voter_registration_status: active/inactive splits. - idx_voter_registration_zip: ZIP-based filtering.

Table: app.election_results

Purpose: Presidential election results by region and year.

Name	Type	Details
region_id	VARCHAR(10) NOT NULL	Region/jurisdiction code (string).
year	INT NOT NULL	Election year.
rep_votes	INT NOT NULL	Republican votes.
dem_votes	INT NOT NULL	Democratic votes.
other_votes	INT	Votes for other parties/candidates.
total_votes	GENERATED ALWAYS AS (rep_votes + dem_votes + COALESCE(other_votes,0)) STORED	

Primary Key & Indexes (what they speed up)

- PRIMARY KEY (region_id, year): one result row per region-year. - idx_election_results_year: filter on election year. - idx_election_results_region_year: drilling into a region's series.

Table: app.cvap_data

Purpose: CVAP demographic estimates by region and year.

Name	Type	Details
region_id	VARCHAR(10) NOT NULL	Region/jurisdiction code (string).
estimate_year	SMALLINT NOT NULL	Year of the estimate.
cvap_total	INT NOT NULL	Total CVAP.
cvap_white	INT	White population.
cvap_black	INT	Black population.
cvap_hispanic	INT	Hispanic population.
cvap_asian	INT	Asian population.
cvap_other	INT	Other population.

Primary Key & Indexes (what they speed up)

- PRIMARY KEY (region_id, estimate_year): uniqueness per region-year estimate. - idx_cvap_data_year: filters by estimate year. - idx_cvap_data_region_year: fast region time series.

Views (verbatim SQL with purposes)

View: v_states_lookup

Purpose: Lightweight directory of states (id, code, name) for dropdowns and joins.

```
CREATE OR REPLACE VIEW app.v_states_lookup AS
SELECT
  s.state_id,
  s.code  AS state_code,
  s.name  AS state_name
FROM app.states s
ORDER BY s.name;
```

View: v_region_year_turnout

Purpose: Turnout rate per region-year: ballots_cast ÷ registered.

```
CREATE OR REPLACE VIEW app.v_region_year_turnout AS
SELECT
  region_id,
  year,
  total_registered,
  total_ballots_cast,
  CASE
    WHEN total_registered > 0
    THEN total_ballots_cast::numeric / total_registered
    ELSE NULL
  END AS turnout_rate
FROM app.eavs_data;
```

View: v_region_year_early_mail_rates

Purpose: Early and mail ballot shares per region-year.

```
CREATE OR REPLACE VIEW app.v_region_year_early_mail_rates AS
SELECT
  d.region_id,
  d.year,
  d.early_voting_total,
  d.ballots_by_mail,
  d.total_ballots_cast,
  CASE WHEN d.total_ballots_cast > 0
       THEN d.early_voting_total::numeric / d.total_ballots_cast
       ELSE NULL END AS early_share,
  CASE WHEN d.total_ballots_cast > 0
       THEN d.ballots_by_mail::numeric / d.total_ballots_cast
       ELSE NULL END AS mail_share
FROM app.eavs_data d;
```

View: v_region_year_mail_rejects

Purpose: Mail ballot rejection rate per region-year (rejected ÷ mail ballots).

```
CREATE OR REPLACE VIEW app.v_region_year_mail_rejects AS
SELECT
  d.region_id,
  d.year,
  d.ballots_by_mail,
  d.mail_reject_total,
  CASE WHEN d.ballots_by_mail > 0
       THEN d.mail_reject_total::numeric / d.ballots_by_mail
       ELSE NULL END AS mail_reject_rate
FROM app.eavs_data d;
```

View: v_region_year_provisional_rates

Purpose: Provisional ballot rate per region-year (provisional ÷ total ballots).

```
CREATE OR REPLACE VIEW app.v_region_year_provisional_rates AS
SELECT
  d.region_id,
  d.year,
  d.prov_cast,
  d.total_ballots_cast,
  CASE WHEN d.total_ballots_cast > 0
       THEN d.prov_cast::numeric / d.total_ballots_cast
       ELSE NULL END AS provisional_rate
FROM app.eavs_data d;
```

View: v_region_year_equipment_summary

Purpose: Total devices per region-year (summed quantities).

```
CREATE OR REPLACE VIEW app.v_region_year_equipment_summary AS
SELECT
  eu.region_id,
  eu.year,
  SUM(eu.quantity) AS total_devices
FROM app.equipment_usage eu
GROUP BY eu.region_id, eu.year;
```

View: v_device_model_usage

Purpose: Footprint of each device model: regions using it and units deployed.

```
CREATE OR REPLACE VIEW app.v_device_model_usage AS
SELECT
  dm.device_model_id,
  dm.vendor,
  dm.model_name,
  dm.device_type,
  dm.year_introduced,
  dm.certification,
  COUNT(DISTINCT eu.region_id) AS regions_using,
  SUM(eu.quantity)             AS units_deployed
FROM app.device_model dm
LEFT JOIN app.equipment_usage eu ON eu.device_model_id = dm.device_model_id
GROUP BY dm.device_model_id, dm.vendor, dm.model_name, dm.device_type, dm.year_introduced, dm.certification;
```

View: v_state_year_summary

Purpose: State-level aggregates of EAVS with turnout, early, and mail shares.

```
CREATE OR REPLACE VIEW app.v_state_year_summary AS
SELECT
  g.state_id,
  s.code      AS state_code,
  s.name      AS state_name,
  d.year,
  SUM(d.total_registered)     AS total_registered,
  SUM(d.total_ballots_cast)  AS total_ballots_cast,
  SUM(d.early_voting_total)  AS early_voting_total,
  SUM(d.ballots_by_mail)      AS ballots_by_mail,
  SUM(d.prov_cast)            AS prov_cast,
  CASE WHEN SUM(d.total_registered) > 0
       THEN SUM(d.total_ballots_cast)::numeric / SUM(d.total_registered)
       ELSE NULL END          AS turnout_rate,
  CASE WHEN SUM(d.total_ballots_cast) > 0
       THEN SUM(d.early_voting_total)::numeric / SUM(d.total_ballots_cast)
       ELSE NULL END          AS early_share,
  CASE WHEN SUM(d.total_ballots_cast) > 0
       THEN SUM(d.ballots_by_mail)::numeric / SUM(d.total_ballots_cast)
       ELSE NULL END          AS mail_share
FROM app.eavs_data d
JOIN app.eavs_geounit g ON g.region_id = d.region_id
JOIN app.states s       ON s.state_id  = g.state_id
GROUP BY g.state_id, s.code, s.name, d.year;
```

View: v_state_year_results

Purpose: State-level aggregates of results with shares, margins, and winner.

```
CREATE OR REPLACE VIEW app.v_state_year_results AS
SELECT
  g.state_id,
  s.code      AS state_code,
  s.name      AS state_name,
  r.year,
  SUM(r.rep_votes)                  AS rep_votes,
  SUM(r.dem_votes)                  AS dem_votes,
  SUM(COALESCE(r.other_votes,0))    AS other_votes,
  SUM(r.total_votes)                AS total_votes,
  CASE WHEN SUM(r.total_votes) > 0
       THEN SUM(r.rep_votes)::numeric / SUM(r.total_votes) ELSE NULL END AS rep_share,
  CASE WHEN SUM(r.total_votes) > 0
       THEN SUM(r.dem_votes)::numeric / SUM(r.total_votes) ELSE NULL END AS dem_share,
  CASE WHEN SUM(r.total_votes) > 0
       THEN SUM(COALESCE(r.other_votes,0))::numeric / SUM(r.total_votes) ELSE NULL END AS other_share,
  (SUM(r.rep_votes) - SUM(r.dem_votes)) AS margin_raw,
  CASE WHEN SUM(r.total_votes) > 0
       THEN (SUM(r.rep_votes) - SUM(r.dem_votes))::numeric / SUM(r.total_votes) ELSE NULL END AS margin_pct,
  CASE
    WHEN SUM(r.rep_votes) > GREATEST(SUM(r.dem_votes), SUM(COALESCE(r.other_votes,0))) THEN 'R'
    WHEN SUM(r.dem_votes) > GREATEST(SUM(r.rep_votes), SUM(COALESCE(r.other_votes,0))) THEN 'D'
    ELSE 'Other'
  END AS winner
FROM app.election_results r
JOIN app.eavs_geounit g ON g.region_id = r.region_id
JOIN app.states s       ON s.state_id  = g.state_id
GROUP BY g.state_id, s.code, s.name, r.year;
```

View: v_eavs_latest_year

Purpose: Most recent EAVS year for default dashboards.

```
CREATE OR REPLACE VIEW app.v_eavs_latest_year AS
WITH last AS (SELECT MAX(year) AS y FROM app.eavs_data)
SELECT d.*
FROM app.eavs_data d
CROSS JOIN last
WHERE d.year = last.y;
```

View: v_results_latest_year

Purpose: Most recent results year for default views.

```
CREATE OR REPLACE VIEW app.v_results_latest_year AS
WITH last AS (SELECT MAX(year) AS y FROM app.election_results)
SELECT r.*
FROM app.election_results r
CROSS JOIN last
WHERE r.year = last.y;
```

View: v_devices_latest_year

Purpose: Most recent equipment usage year for device dashboards.

```
CREATE OR REPLACE VIEW app.v_devices_latest_year AS
WITH last AS (SELECT MAX(year) AS y FROM app.equipment_usage)
SELECT eu.*
FROM app.equipment_usage eu
CROSS JOIN last
WHERE eu.year = last.y;
```

View: v_region_year_basics_lite

Purpose: Lite region-year basics (core metrics without heavy joins).

```
CREATE OR REPLACE VIEW app.v_region_year_basics_lite AS
SELECT
  d.region_id,
  d.state_id,
  s.code  AS state_code,
  s.name  AS state_name,
  d.year,
  d.total_registered,
  d.total_ballots_cast,
  d.ballots_by_mail,
  d.early_voting_total,
  d.prov_cast
FROM app.eavs_data d
JOIN app.states s ON s.state_id = d.state_id;
```

View: v_region_year_results_lite

Purpose: Lite region-year results breakdown (shares, margins).

```
CREATE OR REPLACE VIEW app.v_region_year_results_lite AS
```

```
SELECT
  r.region_id,
  r.state_id,
  s.code AS state_code,
  s.name AS state_name,
  r.year,
  r.rep_votes,
  r.dem_votes,
  COALESCE(r.other_votes, 0) AS other_votes,
  r.total_votes,
  CASE WHEN r.total_votes > 0 THEN r.rep_votes::numeric / r.total_votes ELSE NULL END AS rep_share,
  CASE WHEN r.total_votes > 0 THEN r.dem_votes::numeric / r.total_votes ELSE NULL END AS dem_share,
  CASE WHEN r.total_votes > 0 THEN COALESCE(r.other_votes,0)::numeric / r.total_votes ELSE NULL END AS other_share,
  (r.rep_votes - r.dem_votes) AS margin_raw,
  CASE WHEN r.total_votes > 0 THEN (r.rep_votes - r.dem_votes)::numeric / r.total_votes ELSE NULL END AS margin_pct,
  CASE
    WHEN r.rep_votes > GREATEST(r.dem_votes, COALESCE(r.other_votes,0)) THEN 'R'
    WHEN r.dem_votes > GREATEST(r.rep_votes, COALESCE(r.other_votes,0)) THEN 'D'
    ELSE 'Other'
  END AS winner
FROM app.election_results r
JOIN app.states s ON s.state_id = r.state_id;
```

View: v_region_year_equipment_lite

Purpose: Lite region-year equipment overview with key fields.

```
CREATE OR REPLACE VIEW app.v_region_year_equipment_lite AS
SELECT
  eu.region_id,
  eu.state_id,
  s.code AS state_code,
  s.name AS state_name,
  eu.year,
  eu.device_model_id,
  dm.vendor,
  dm.model_name,
  dm.device_type,
  eu.quantity
FROM app.equipment_usage eu
JOIN app.states s        ON s.state_id  = eu.state_id
JOIN app.device_model dm ON dm.device_model_id = eu.device_model_id;
```

View: v_region_year_cvap_lite

Purpose: Lite region-year CVAP demographics for quick charts.

```
CREATE OR REPLACE VIEW app.v_region_year_cvap_lite AS
SELECT
  c.region_id,
  c.state_id,
  s.code AS state_code,
  s.name AS state_name,
  c.estimate_year AS year,
  c.cvap_total,
  c.cvap_white,
  c.cvap_black,
  c.cvap_hispanic,
  c.cvap_asian,
  c.cvap_other
FROM app.cvap_data c
JOIN app.states s ON s.state_id = c.state_id;
```