```
CREATE TABLE stg_pan_number_dataset
(
pan_number   text
);

SELECT * FROM stg_pan_number_dataset;

--Identify and handle missing data

SELECT * FROM stg_pan_number_dataset WHERE
pan_number IS NULL;

--Check for duplicates

SELECT pan_number, COUNT(1)
FROM stg_pan_number_dataset
GROUP BY pan_number
HAVING COUNT(1) > 1;

--Handle leading/trailing spaces

SELECT * FROM stg_pan_number_dataset WHERE
pan_number <> TRIM(pan_number);

--Correct letter case

SELECT * FROM stg_pan_number_dataset WHERE
pan_number <> UPPER(pan_number);

--Cleaned pan numbers

SELECT DISTINCT UPPER(TRIM(pan_number)) AS
pan_number
```

```
FROM stg_pan_number_dataset
WHERE pan_number IS NOT NULL
AND TRIM(pan_number) <> '';


--Function to check if the adjacent characters
are the same

CREATE OR REPLACE FUNCTION fn_check(p_str text)
RETURNS boolean
language plpgsql
AS $$
BEGIN
     FOR i in 1 .. (length(p_str) - 1)
         loop
         if substring(p_str, i ,1) =
substring(p_str, i+1 ,1)
         then
         return true; --characters are adjacent
         end if;
         end loop;
         return false; --none of the characters
adjacent to each other are the same
end;
$$

SELECT fn_check('ABDRC')


--Function to check if the sequential characters
are used

CREATE OR REPLACE FUNCTION fn_check_seq(p_str
text)
RETURNS boolean
language plpgsql
AS $$
BEGIN
     FOR i in 1 .. (length(p_str) - 1)
         loop
         if ascii(substring(p_str, i+1 ,1)) -
ascii(substring(p_str, i ,1)) <> 1
         then
         return false; -- No sequence
         end if;
         end loop;
         return true;   --THE STRING is forming
a sequence
end;
$$

SELECT fn_check_seq('ABGDE')

--Regular expression to validate the pattern or
structure of PAN numbers

SELECT *
 FROM stg_pan_number_dataset
 WHERE pan_number ~ '^[A-Z]{5}[0-9]{4}[A-Z]$'


--Valid and Invalid pan categorization

WITH cte_cleaned_pan AS (
      SELECT DISTINCT UPPER(TRIM(pan_number)) AS
pan_number
```

```
      FROM stg_pan_number_dataset
      WHERE pan_number IS NOT NULL
      AND TRIM(pan_number) <> '')
SELECT * FROM cte_cleaned_pan
WHERE fn_check(pan_number) = false
AND fn_check_seq(substring(pan_number,1,5)) =
false
AND fn_check_seq(substring(pan_number,6,4)) =
false
AND pan_number ~ '^[A-Z]{5}[0-9]{4}[A-Z]$'


--Final query

WITH cte_cleaned_pan AS (
      SELECT DISTINCT UPPER(TRIM(pan_number)) AS
pan_number
      FROM stg_pan_number_dataset
      WHERE pan_number IS NOT NULL
      AND TRIM(pan_number) <> ''),
        cte_valid_pans AS
      (SELECT * FROM cte_cleaned_pan
      WHERE fn_check(pan_number) = false
      AND
fn_check_seq(substring(pan_number,1,5)) = false
      AND
fn_check_seq(substring(pan_number,6,4)) = false
      AND pan_number ~
'^[A-Z]{5}[0-9]{4}[A-Z]$')
SELECT cln.pan_number,
       CASE WHEN vld.pan_number IS NOT NULL
                      THEN 'VALID PAN'
                         ELSE 'INVALID PAN'
        END AS status
FROM cte_cleaned_pan cln
LEFT JOIN cte_valid_pans vld ON vld.pan_number =
cln.pan_number;


--Creating a view to reuse this final query and
generating a summary report which will generate
following results

CREATE OR REPLACE VIEW vw_valid_invalid_pans
AS
WITH cte_cleaned_pan AS (
      SELECT DISTINCT UPPER(TRIM(pan_number)) AS
pan_number
      FROM stg_pan_number_dataset
      WHERE pan_number IS NOT NULL
      AND TRIM(pan_number) <> ''),
        cte_valid_pans AS
      (SELECT * FROM cte_cleaned_pan
      WHERE fn_check(pan_number) = false
      AND
fn_check_seq(substring(pan_number,1,5)) = false
      AND
fn_check_seq(substring(pan_number,6,4)) = false
      AND pan_number ~
'^[A-Z]{5}[0-9]{4}[A-Z]$')
SELECT cln.pan_number,
       CASE WHEN vld.pan_number IS NOT NULL
                     THEN 'VALID PAN'
                        ELSE 'INVALID PAN'
        END AS status
FROM cte_cleaned_pan cln
```

```
LEFT JOIN cte_valid_pans vld ON vld.pan_number =
cln.pan_number;

SELECT * FROM vw_valid_invalid_pans;


--Count of how many are valid and how many are
invalid pans and total processed records
SELECT
(SELECT COUNT(*) FROM stg_pan_number_dataset) AS
total_processed_records,
COUNT(*) FILTER(WHERE STATUS = 'VALID PAN') AS
total_valid_pans,
COUNT(*) FILTER(WHERE STATUS = 'INVALID PAN') AS
total_invalid_pans
FROM vw_valid_invalid_pans;

--Total missing pans

WITH cte AS (
SELECT
(SELECT COUNT(*) FROM stg_pan_number_dataset) AS
total_processed_records,
COUNT(*) FILTER(WHERE STATUS = 'VALID PAN') AS
total_valid_pans,
COUNT(*) FILTER(WHERE STATUS = 'INVALID PAN') AS
total_invalid_pans
FROM vw_valid_invalid_pans
)
SELECT total_processed_records,
total_valid_pans, total_invalid_pans,
       (total_processed_records -
(total_valid_pans + total_invalid_pans)) AS
```

```
total_missing_pans
FROM cte;
```