**SQL ASSIGNMENT**

***TASK -1: Generate a random data input file***

*a) Feed-1, which has 10 columns with 10 rows,*

*b) Feed-2, which has 15 columns with 15 rows,*

*c) Feed-3, which has 20 columns with 20 rows*

----------------------------------------------------------------------------------------------------------------------------------------------

**FEED-1 QUERY AND OUTPUT BELOW**

CREATE TABLE feed1 (

col\_1 INT,

col\_2 VARCHAR(255),

col\_3 VARCHAR(255),

col\_4 VARCHAR(255),

col\_5 VARCHAR(255),

col\_6 VARCHAR(255),

col\_7 VARCHAR(255),

col\_8 VARCHAR(255),

col\_9 VARCHAR(255),

col\_10 VARCHAR(255)

);

-- Insert 10 random rows

INSERT INTO feed1 (col\_1, col\_2, col\_3, col\_4, col\_5, col\_6, col\_7, col\_8, col\_9, col\_10)

SELECT

g.n,

substr(md5(random()::text), 1, 8),

substr(md5(random()::text), 1, 8),

substr(md5(random()::text), 1, 8),

substr(md5(random()::text), 1, 8),

substr(md5(random()::text), 1, 8),

substr(md5(random()::text), 1, 8),

substr(md5(random()::text), 1, 8),

substr(md5(random()::text), 1, 8),

substr(md5(random()::text), 1, 8)

FROM generate\_series(1,10) AS g(n);

-- Duplicate one row (take the first row and insert again)

INSERT INTO feed1

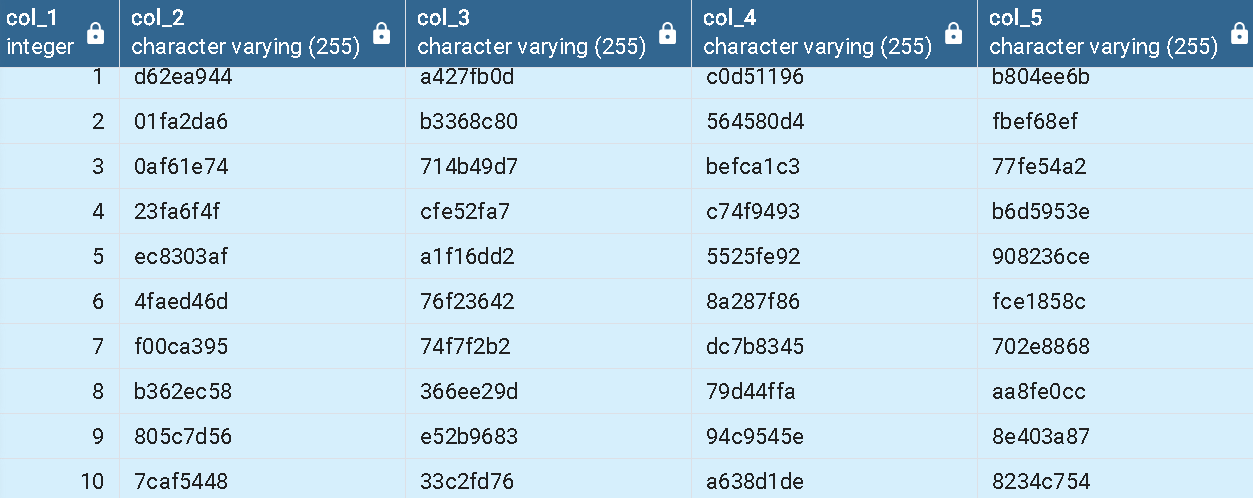
SELECT \*

FROM feed1

OFFSET 0 LIMIT 1;

select \* from feed1 ;

OUTPUT BELOW



--------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**FEED-2 QUERY AND OUTPUT BELOW**

INSERT INTO Feed2 (col\_1, col\_2, col\_3, col\_4, col\_5, col\_6, col\_7, col\_8, col\_9, col\_10,

col\_11, col\_12, col\_13, col\_14, col\_15)

SELECT

g.n,

SUBSTRING(MD5(RANDOM()::text), 1, 8),

SUBSTRING(MD5(RANDOM()::text), 1, 8),

SUBSTRING(MD5(RANDOM()::text), 1, 8),

SUBSTRING(MD5(RANDOM()::text), 1, 8),

SUBSTRING(MD5(RANDOM()::text), 1, 8),

SUBSTRING(MD5(RANDOM()::text), 1, 8),

SUBSTRING(MD5(RANDOM()::text), 1, 8),

SUBSTRING(MD5(RANDOM()::text), 1, 8),

SUBSTRING(MD5(RANDOM()::text), 1, 8),

SUBSTRING(MD5(RANDOM()::text), 1, 8),

SUBSTRING(MD5(RANDOM()::text), 1, 8),

SUBSTRING(MD5(RANDOM()::text), 1, 8),

SUBSTRING(MD5(RANDOM()::text), 1, 8),

SUBSTRING(MD5(RANDOM ()::text), 1, 8)

FROM generate\_series(1, 15) g(n);

-- Duplicate first 2 rows (like LIMIT 2 in MySQL)

INSERT INTO Feed2

SELECT \* FROM Feed2

WHERE col\_1 IN (1, 2);

SELECT \* FROM FEED2

DROP TABLE IF EXISTS Feed3;

CREATE TABLE Feed3 (

col\_1 INT,

col\_2 VARCHAR(255), col\_3 VARCHAR(255), col\_4 VARCHAR(255), col\_5 VARCHAR(255),

col\_6 VARCHAR(255), col\_7 VARCHAR(255), col\_8 VARCHAR(255), col\_9 VARCHAR(255), col\_10 VARCHAR(255),

col\_11 VARCHAR(255), col\_12 VARCHAR(255), col\_13 VARCHAR(255), col\_14 VARCHAR(255), col\_15 VARCHAR(255),

col\_16 VARCHAR(255), col\_17 VARCHAR(255), col\_18 VARCHAR(255), col\_19 VARCHAR(255), col\_20 VARCHAR(255)

);

-- Insert 20 rows with random values

INSERT INTO Feed3 (col\_1, col\_2, col\_3, col\_4, col\_5, col\_6, col\_7, col\_8, col\_9, col\_10,

col\_11, col\_12, col\_13, col\_14, col\_15, col\_16, col\_17, col\_18, col\_19, col\_20)

SELECT

g.n,

SUBSTRING(MD5(RANDOM()::text), 1, 8), -- col\_2

SUBSTRING(MD5(RANDOM()::text), 1, 8), -- col\_3

SUBSTRING(MD5(RANDOM()::text), 1, 8), -- col\_4

SUBSTRING(MD5(RANDOM()::text), 1, 8), -- col\_5

SUBSTRING(MD5(RANDOM()::text), 1, 8), -- col\_6

SUBSTRING(MD5(RANDOM()::text), 1, 8), -- col\_7

SUBSTRING(MD5(RANDOM()::text), 1, 8), -- col\_8

SUBSTRING(MD5(RANDOM()::text), 1, 8), -- col\_9

SUBSTRING(MD5(RANDOM()::text), 1, 8), -- col\_10

SUBSTRING(MD5(RANDOM()::text), 1, 8), -- col\_11

SUBSTRING(MD5(RANDOM()::text), 1, 8), -- col\_12

SUBSTRING(MD5(RANDOM()::text), 1, 8), -- col\_13

SUBSTRING(MD5(RANDOM()::text), 1, 8), -- col\_14

SUBSTRING(MD5(RANDOM()::text), 1, 8), -- col\_15

SUBSTRING(MD5(RANDOM()::text), 1, 8), -- col\_16

SUBSTRING(MD5(RANDOM()::text), 1, 8), -- col\_17

SUBSTRING(MD5(RANDOM()::text), 1, 8), -- col\_18

SUBSTRING(MD5(RANDOM()::text), 1, 8), -- col\_19

SUBSTRING(MD5(RANDOM()::text), 1, 8) -- col\_20

FROM generate\_series(1, 20) g(n);

INSERT INTO Feed3 (col\_1, col\_2, col\_3, col\_4, col\_5, col\_6, col\_7, col\_8, col\_9, col\_10,

col\_11, col\_12, col\_13, col\_14, col\_15, col\_16, col\_17, col\_18, col\_19, col\_20)

SELECT

(SELECT MAX(col\_1) FROM Feed3) + ROW\_NUMBER() OVER(),

col\_2, col\_3, col\_4, col\_5, col\_6, col\_7, col\_8, col\_9, col\_10,

col\_11, col\_12, col\_13, col\_14, col\_15, col\_16, col\_17, col\_18, col\_19, col\_20

FROM Feed3

WHERE col\_1 IN (1, 2);

SELECT \* FROM FEED3

-----------------------------------------------------------------------------------------------------------------------------------------------------------------------

**TASK -2**

***Automate the Rea 1 input file generation using SQL scripts, and the parameter will be "Feed name\* & Number of Rows to populate Data***

CREATE OR REPLACE PROCEDURE generate\_feed(

table\_name TEXT,

num\_cols INT,

num\_rows INT

)

LANGUAGE plpgsql

AS $$

DECLARE

i INT;

j INT;

create\_table\_sql TEXT;

col\_list TEXT := '';

val\_list TEXT;

BEGIN

-- Drop table if exists

EXECUTE format('DROP TABLE IF EXISTS %I', table\_name);

-- Build CREATE TABLE dynamically

create\_table\_sql := 'CREATE TABLE ' || quote\_ident(table\_name) || ' (';

FOR i IN 1..num\_cols LOOP

create\_table\_sql := create\_table\_sql || 'col\_' || i || ' VARCHAR(255)';

IF i < num\_cols THEN

create\_table\_sql := create\_table\_sql || ', ';

END IF;

END LOOP;

create\_table\_sql := create\_table\_sql || ')';

EXECUTE create\_table\_sql;

-- Build column list

col\_list := '';

FOR i IN 1..num\_cols LOOP

col\_list := col\_list || 'col\_' || i;

IF i < num\_cols THEN

col\_list := col\_list || ', ';

END IF;

END LOOP;

-- Insert rows

FOR j IN 1..num\_rows LOOP

val\_list := '';

FOR i IN 1..num\_cols LOOP

val\_list := val\_list || quote\_literal(substr(md5(random()::text), 1, 8));

IF i < num\_cols THEN

val\_list := val\_list || ', ';

END IF;

END LOOP;

EXECUTE 'INSERT INTO ' || quote\_ident(table\_name) || '(' || col\_list || ') VALUES (' || val\_list || ')';

END LOOP;

-- Duplicate first row if rows exist

IF num\_rows > 0 THEN

EXECUTE 'INSERT INTO ' || quote\_ident(table\_name) || '(' || col\_list || ') ' ||

'SELECT ' || col\_list || ' FROM ' || quote\_ident(table\_name) || ' LIMIT 1';

END IF;

RAISE NOTICE 'Table % created and populated with % rows and % columns.', table\_name, num\_rows, num\_cols;

END;

$$;

**CALL generate\_feed('feed3', 20, 20);**

**REQ 3,4 Write SQL script to identify the duplicate (rows) in each of the table Feed-1, 2, 3**

**Write the duplicate records in output file - "duplicates"**

CREATE TABLE IF NOT EXISTS duplicates (

table\_name\_source VARCHAR(255),

count\_of\_duplicates INT,

col\_1 VARCHAR(255), col\_2 VARCHAR(255), col\_3 VARCHAR(255), col\_4 VARCHAR(255), col\_5 VARCHAR(255),

col\_6 VARCHAR(255), col\_7 VARCHAR(255), col\_8 VARCHAR(255), col\_9 VARCHAR(255), col\_10 VARCHAR(255),

col\_11 VARCHAR(255), col\_12 VARCHAR(255), col\_13 VARCHAR(255), col\_14 VARCHAR(255), col\_15 VARCHAR(255),

col\_16 VARCHAR(255), col\_17 VARCHAR(255), col\_18 VARCHAR(255), col\_19 VARCHAR(255), col\_20 VARCHAR(255)

);

TRUNCATE TABLE duplicates;

INSERT INTO duplicates (

table\_name\_source, count\_of\_duplicates,

col\_1, col\_2, col\_3, col\_4, col\_5, col\_6, col\_7, col\_8, col\_9, col\_10

)

SELECT

'Feed1' AS table\_name\_source, COUNT(\*) AS count\_of\_duplicates,

col\_1, col\_2, col\_3, col\_4, col\_5, col\_6, col\_7, col\_8, col\_9, col\_10

FROM Feed1

GROUP BY col\_1, col\_2, col\_3, col\_4, col\_5, col\_6, col\_7, col\_8, col\_9, col\_10

HAVING COUNT(\*) > 1;

INSERT INTO duplicates (

table\_name\_source, count\_of\_duplicates,

col\_1, col\_2, col\_3, col\_4, col\_5, col\_6, col\_7, col\_8, col\_9, col\_10,

col\_11, col\_12, col\_13, col\_14, col\_15

)

SELECT

'Feed2' AS table\_name\_source, COUNT(\*) AS count\_of\_duplicates,

col\_1, col\_2, col\_3, col\_4, col\_5, col\_6, col\_7, col\_8, col\_9, col\_10,

col\_11, col\_12, col\_13, col\_14, col\_15

FROM Feed2

GROUP BY col\_1, col\_2, col\_3, col\_4, col\_5, col\_6, col\_7, col\_8, col\_9, col\_10,

col\_11, col\_12, col\_13, col\_14, col\_15

HAVING COUNT(\*) > 1;

INSERT INTO duplicates (

table\_name\_source, count\_of\_duplicates,

col\_1, col\_2, col\_3, col\_4, col\_5, col\_6, col\_7, col\_8, col\_9, col\_10,

col\_11, col\_12, col\_13, col\_14, col\_15, col\_16, col\_17, col\_18, col\_19, col\_20

)

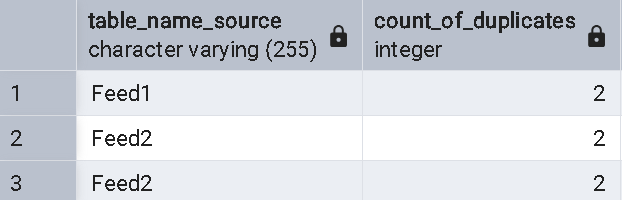
SELECT

'Feed3' AS table\_name\_source, COUNT(\*) AS count\_of\_duplicates,

col\_1, col\_2, col\_3, col\_4, col\_5, col\_6, col\_7, col\_8, col\_9, col\_10,

col\_11, col\_12, col\_13, col\_14, col\_15, col\_16, col\_17, col\_18, col\_19, col\_20

FROM Feed3

GROUP BY col\_1, col\_2, col\_3, col\_4, col\_5, col\_6, col\_7, col\_8, col\_9, col\_10,

col\_11, col\_12, col\_13, col\_14, col\_15, col\_16, col\_17, col\_18, col\_19, col\_20

HAVING COUNT(\*) > 1;

SELECT \* FROM duplicates;

***REQ – 5 Create a script to replace all the duplicates with Unique rows and update back to the respective Feed table***

DELETE FROM Feed1 a

USING (

SELECT MIN(ctid) AS keep\_ctid, col\_1, col\_2, col\_3, col\_4, col\_5,

col\_6, col\_7, col\_8, col\_9, col\_10

FROM Feed1

GROUP BY col\_1, col\_2, col\_3, col\_4, col\_5,

col\_6, col\_7, col\_8, col\_9, col\_10

HAVING COUNT(\*) > 1

) b

WHERE a.col\_1 = b.col\_1 AND a.col\_2 = b.col\_2 AND a.col\_3 = b.col\_3 AND a.col\_4 = b.col\_4

AND a.col\_5 = b.col\_5 AND a.col\_6 = b.col\_6 AND a.col\_7 = b.col\_7 AND a.col\_8 = b.col\_8

AND a.col\_9 = b.col\_9 AND a.col\_10 = b.col\_10

AND a.ctid <> b.keep\_ctid;

-- For Feed2 (15 columns)

DELETE FROM Feed2 a

USING (

SELECT MIN(ctid) AS keep\_ctid, col\_1, col\_2, col\_3, col\_4, col\_5,

col\_6, col\_7, col\_8, col\_9, col\_10,

col\_11, col\_12, col\_13, col\_14, col\_15

FROM Feed2

GROUP BY col\_1, col\_2, col\_3, col\_4, col\_5,

col\_6, col\_7, col\_8, col\_9, col\_10,

col\_11, col\_12, col\_13, col\_14, col\_15

HAVING COUNT(\*) > 1

) b

WHERE a.col\_1 = b.col\_1 AND a.col\_2 = b.col\_2 AND a.col\_3 = b.col\_3 AND a.col\_4 = b.col\_4

AND a.col\_5 = b.col\_5 AND a.col\_6 = b.col\_6 AND a.col\_7 = b.col\_7 AND a.col\_8 = b.col\_8

AND a.col\_9 = b.col\_9 AND a.col\_10 = b.col\_10 AND a.col\_11 = b.col\_11 AND a.col\_12 = b.col\_12

AND a.col\_13 = b.col\_13 AND a.col\_14 = b.col\_14 AND a.col\_15 = b.col\_15

AND a.ctid <> b.keep\_ctid;

-- For Feed3 (20 columns)

DELETE FROM Feed3 a

USING (

SELECT MIN(ctid) AS keep\_ctid, col\_1, col\_2, col\_3, col\_4, col\_5,

col\_6, col\_7, col\_8, col\_9, col\_10,

col\_11, col\_12, col\_13, col\_14, col\_15,

col\_16, col\_17, col\_18, col\_19, col\_20

FROM Feed3

GROUP BY col\_1, col\_2, col\_3, col\_4, col\_5,

col\_6, col\_7, col\_8, col\_9, col\_10,

col\_11, col\_12, col\_13, col\_14, col\_15,

col\_16, col\_17, col\_18, col\_19, col\_20

HAVING COUNT(\*) > 1

) b

WHERE a.col\_1 = b.col\_1 AND a.col\_2 = b.col\_2 AND a.col\_3 = b.col\_3 AND a.col\_4 = b.col\_4

AND a.col\_5 = b.col\_5 AND a.col\_6 = b.col\_6 AND a.col\_7 = b.col\_7 AND a.col\_8 = b.col\_8

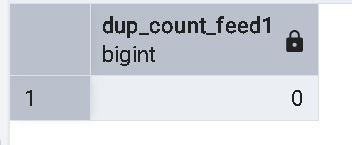
AND a.col\_9 = b.col\_9 AND a.col\_10 = b.col\_10 AND a.col\_11 = b.col\_11 AND a.col\_12 = b.col\_12

AND a.col\_13 = b.col\_13 AND a.col\_14 = b.col\_14 AND a.col\_15 = b.col\_15 AND a.col\_16 = b.col\_16

AND a.col\_17 = b.col\_17 AND a.col\_18 = b.col\_18 AND a.col\_19 = b.col\_19 AND a.col\_20 = b.col\_20

AND a.ctid <> b.keep\_ctid;

--------------------------------------------------------------------------------------------------------------------------------------------------------------------------

***REQ – 6 Execute the duplicate script and check the output is zero***

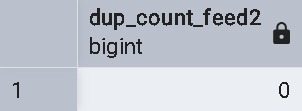
SELECT COUNT(\*) AS dup\_count\_feed1

FROM (

SELECT COUNT(\*)

FROM Feed1

GROUP BY col\_1, col\_2, col\_3, col\_4, col\_5,

 col\_6, col\_7, col\_8, col\_9, col\_10

HAVING COUNT(\*) > 1

) t;

SELECT COUNT(\*) AS dup\_count\_feed2

FROM (

SELECT COUNT(\*)

FROM Feed2

GROUP BY col\_1, col\_2, col\_3, col\_4, col\_5,

col\_6, col\_7, col\_8, col\_9, col\_10,

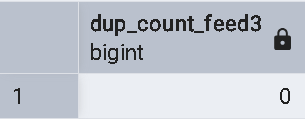
col\_11, col\_12, col\_13, col\_14, col\_15

HAVING COUNT(\*) > 1

) t;

SELECT COUNT(\*) AS dup\_count\_feed3

FROM (

 SELECT COUNT(\*)

FROM Feed3

GROUP BY col\_1, col\_2, col\_3, col\_4, col\_5,

col\_6, col\_7, col\_8, col\_9, col\_10,

col\_11, col\_12, col\_13, col\_14, col\_15,

col\_16, col\_17, col\_18, col\_19, col\_20

HAVING COUNT(\*) > 1

) t;

-----------------------------------------------------------------------------------------------------------------------------------------------------------------------

***Req – 7 Create SOL script to compare data from Feed-2,3 to Feed-1 and write in***

***output file on the compared results***

CREATE TABLE IF NOT EXISTS comparison\_result (

source\_feed VARCHAR(255),

target\_feed VARCHAR(255),

record\_data\_col1 VARCHAR(255),

comparison\_status VARCHAR(255)

);

-- Clear old results

TRUNCATE TABLE comparison\_result;

-- Compare Feed2 to Feed1

INSERT INTO comparison\_result (source\_feed, target\_feed, record\_data\_col1, comparison\_status)

SELECT 'Feed2', 'Feed1', s.col\_1, 'In source only'

FROM Feed2 s LEFT JOIN Feed1 t ON s.col\_1 = t.col\_1

WHERE t.col\_1 IS NULL;

INSERT INTO comparison\_result (source\_feed, target\_feed, record\_data\_col1, comparison\_status)

SELECT 'Feed2', 'Feed1', t.col\_1, 'In target only'

FROM Feed1 t LEFT JOIN Feed2 s ON t.col\_1 = s.col\_1

WHERE s.col\_1 IS NULL;

-- Compare Feed3 to Feed1

INSERT INTO comparison\_result (source\_feed, target\_feed, record\_data\_col1, comparison\_status)

SELECT 'Feed3', 'Feed1', s.col\_1, 'In source only'

FROM Feed3 s LEFT JOIN Feed1 t ON s.col\_1 = t.col\_1

WHERE t.col\_1 IS NULL;

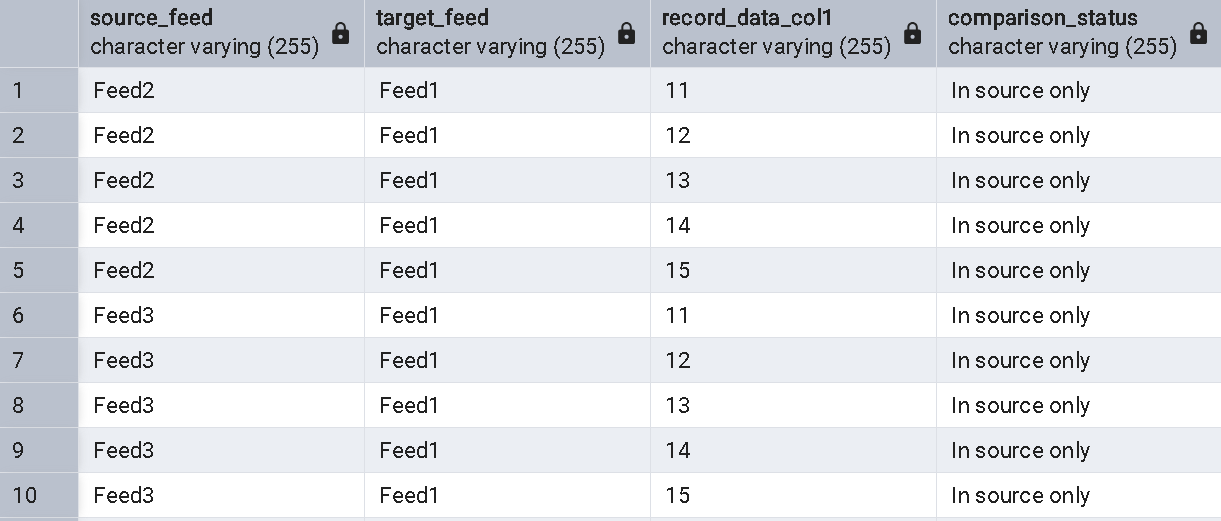
INSERT INTO comparison\_result (source\_feed, target\_feed, record\_data\_col1, comparison\_status)

SELECT 'Feed3', 'Feed1', t.col\_1, 'In target only'

FROM Feed1 t LEFT JOIN Feed3 s ON t.col\_1 = s.col\_1

WHERE s.col\_1 IS NULL;

SELECT \* FROM comparison\_result;



--------------------------------------------------------------------------------------------------------------------------------------------------------------------------

***Req – 8,9***

***Create a Test plan with all kinds of manual test cases to test this End***

***functionality***

***Automate the test cases (if possible) using any method, but it should be***

***Automated***

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID** | **Test Scenario** | **Test Steps** | **Expected Result** |
| TC-01 | Validate Feed Table Creation | Run the SQL script for Feed-1, Feed-2, Feed-3 | Tables are created with correct number of columns & rows |
| TC-02 | Validate Data Generation Automation | Execute procedure with parameters (Feed-1, 10 rows) | Data is populated with random values in correct structure |
| TC-03 | **Validate Duplicate Identification** | Execute duplicate identification script | File "duplicates" contains the expected duplicate rows |
| TC-04 | Validate Duplicate Output File | Execute duplicate identification script | File "duplicates" contains the expected duplicate rows |
| TC-05 | Validate Duplicate Removal | Run deduplication script on Feed-1 | Duplicates are removed, only unique rows remain |
| TC-06 | Verify No Duplicates Remain | Run duplicate identification script again | Query returns **zero duplicates** |
| TC-07 | Validate Data Comparison | Run comparison script between Feed-2/Feed-3 with Feed-1 | Output file contains mismatched and matched rows |
| TC-08 | End-to-End Test | Run all scripts in sequence: generate feeds → detect duplicates → remove → compare | **Entire flow works correctly without error** |
| TC-09 | Automation of task |  |  |
| TC-10 | Document the process |  |  |

Req-9

DELIMITER $$

CREATE PROCEDURE generate\_feed(

IN table\_name VARCHAR(255),

IN num\_cols INT,

IN num\_rows INT

)

BEGIN

DECLARE i INT DEFAULT 1;

DECLARE j INT DEFAULT 1;

DECLARE create\_table\_sql TEXT;

DECLARE insert\_sql TEXT;

DECLARE col\_list TEXT;

DECLARE val\_list TEXT;

SET @drop\_sql = CONCAT('DROP TABLE IF EXISTS ', table\_name);

PREPARE stmt FROM @drop\_sql;

EXECUTE stmt;

DEALLOCATE PREPARE stmt;

SET create\_table\_sql = CONCAT('CREATE TABLE ', table\_name, ' (');

WHILE i <= num\_cols DO

SET create\_table\_sql = CONCAT(create\_table\_sql, 'col\_', i, ' VARCHAR(255)');

IF i < num\_cols THEN

SET create\_table\_sql = CONCAT(create\_table\_sql, ', ');

END IF;

SET i = i + 1;

END WHILE;

SET create\_table\_sql = CONCAT(create\_table\_sql, ')');

SET @create\_sql = create\_table\_sql;

PREPARE stmt FROM @create\_sql;

EXECUTE stmt;

DEALLOCATE PREPARE stmt;

SET i = 1;

SET col\_list = '';

WHILE i <= num\_cols DO

SET col\_list = CONCAT(col\_list, 'col\_', i);

IF i < num\_cols THEN

SET col\_list = CONCAT(col\_list, ', ');

END IF;

SET i = i + 1;

END WHILE;

SET j = 1;

WHILE j <= num\_rows DO

SET i = 1;

SET val\_list = '';

WHILE i <= num\_cols DO

SET val\_list = CONCAT(val\_list, '''', SUBSTRING(MD5(RAND()), 1, 8), '''');

IF i < num\_cols THEN

SET val\_list = CONCAT(val\_list, ', ');

END IF;

SET i = i + 1;

END WHILE;

SET @insert\_sql = CONCAT('INSERT INTO ', table\_name, ' (', col\_list, ') VALUES (', val\_list, ')');

PREPARE stmt FROM @insert\_sql;

EXECUTE stmt;

DEALLOCATE PREPARE stmt;

SET j = j + 1;

END WHILE;

IF num\_rows > 0 THEN

SET @insert\_sql = CONCAT('INSERT INTO ', table\_name, ' (', col\_list, ') SELECT ', col\_list, ' FROM ', table\_name, ' LIMIT 1');

PREPARE stmt FROM @insert\_sql;

EXECUTE stmt;

DEALLOCATE PREPARE stmt;

END IF;

SELECT CONCAT('Table ', table\_name, ' created and populated with ', num\_rows, ' rows and ', num\_cols, ' columns.') AS Status;

END$$

DELIMITER ;

CREATE TABLE IF NOT EXISTS duplicates (

table\_name\_source VARCHAR(255),

duplicate\_data JSON,

count\_of\_duplicates INT

);

DELIMITER $$

CREATE PROCEDURE find\_and\_store\_duplicates(IN source\_table\_name VARCHAR(255))

BEGIN

DECLARE done INT DEFAULT FALSE;

DECLARE cols TEXT;

DECLARE cur CURSOR FOR

SELECT GROUP\_CONCAT(COLUMN\_NAME)

FROM INFORMATION\_SCHEMA.COLUMNS

WHERE TABLE\_NAME = source\_table\_name AND TABLE\_SCHEMA = DATABASE();

DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

OPEN cur;

FETCH cur INTO cols;

CLOSE cur;

SET @sql = CONCAT(

'INSERT INTO duplicates (table\_name\_source, duplicate\_data, count\_of\_duplicates) ',

'SELECT ''', source\_table\_name, ''', ',

'JSON\_OBJECT(',

(SELECT GROUP\_CONCAT(CONCAT('''', COLUMN\_NAME, ''', ', COLUMN\_NAME)) FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_NAME = source\_table\_name AND TABLE\_SCHEMA = DATABASE()),

'), COUNT(\*) ',

'FROM ', source\_table\_name, ' ',

'GROUP BY ', cols, ' ',

'HAVING COUNT(\*) > 1'

);

PREPARE stmt FROM @sql;

EXECUTE stmt;

DEALLOCATE PREPARE stmt;

SELECT CONCAT('Duplicate check for table ', source\_table\_name, ' is complete.') AS Status;

END$$

CREATE PROCEDURE replace\_duplicates\_with\_unique(IN target\_table\_name VARCHAR(255))

BEGIN

DECLARE cols TEXT;

SELECT GROUP\_CONCAT(COLUMN\_NAME)

INTO cols

FROM INFORMATION\_SCHEMA.COLUMNS

WHERE TABLE\_NAME = target\_table\_name AND TABLE\_SCHEMA = DATABASE();

SET @temp\_table\_sql = CONCAT('CREATE TEMPORARY TABLE temp\_distinct AS SELECT DISTINCT \* FROM ', target\_table\_name);

PREPARE stmt FROM @temp\_table\_sql;

EXECUTE stmt;

DEALLOCATE PREPARE stmt;

SET @truncate\_sql = CONCAT('TRUNCATE TABLE ', target\_table\_name);

PREPARE stmt FROM @truncate\_sql;

EXECUTE stmt;

DEALLOCATE PREPARE stmt;

SET @insert\_sql = CONCAT('INSERT INTO ', target\_table\_name, ' SELECT \* FROM temp\_distinct');

PREPARE stmt FROM @insert\_sql;

EXECUTE stmt;

DEALLOCATE PREPARE stmt;

DROP TEMPORARY TABLE temp\_distinct;

SELECT CONCAT('Duplicates removed from ', target\_table\_name, '. The table now contains only unique rows.') AS Status;

END$$

CREATE PROCEDURE verify\_no\_duplicates(IN table\_to\_check VARCHAR(255))

BEGIN

DECLARE duplicate\_count INT;

DECLARE cols TEXT;

SELECT GROUP\_CONCAT(COLUMN\_NAME)

INTO cols

FROM INFORMATION\_SCHEMA.COLUMNS

WHERE TABLE\_NAME = table\_to\_check AND TABLE\_SCHEMA = DATABASE();

SET @sql = CONCAT(

'SELECT COUNT(\*) INTO @d\_count FROM (',

'SELECT ', cols, ', COUNT(\*) as cnt ',

'FROM ', table\_to\_check, ' ',

'GROUP BY ', cols, ' ',

'HAVING COUNT(\*) > 1) AS duplicates'

);

PREPARE stmt FROM @sql;

EXECUTE stmt;

DEALLOCATE PREPARE stmt;

SET duplicate\_count = @d\_count;

IF duplicate\_count = 0 THEN

SELECT CONCAT('Verification successful for ', table\_to\_check, ': No duplicate rows found.') AS Result;

ELSE

SELECT CONCAT('Verification failed for ', table\_to\_check, ': ', duplicate\_count, ' duplicate sets found.') AS Result;

END IF;

END$$

CREATE TABLE IF NOT EXISTS comparison\_results (

source\_feed VARCHAR(255),

target\_feed VARCHAR(255),

record\_data JSON,

comparison\_status VARCHAR(255) -- e.g., 'In source only', 'In target only', 'Mismatch'

);

CREATE PROCEDURE compare\_feeds(

IN source\_feed\_name VARCHAR(255),

IN target\_feed\_name VARCHAR(255)

)

BEGIN

DECLARE source\_cols TEXT;

DECLARE target\_cols TEXT;

SET @source\_col1 = 's.col\_1';

SET @target\_col1 = 't.col\_1';

DELETE FROM comparison\_results WHERE source\_feed = source\_feed\_name AND target\_feed = target\_feed\_name;

SET @source\_only\_sql = CONCAT(

'INSERT INTO comparison\_results (source\_feed, target\_feed, record\_data, comparison\_status) ',

'SELECT ''', source\_feed\_name, ''', ''', target\_feed\_name, ''', ',

'JSON\_OBJECT(',

(SELECT GROUP\_CONCAT(CONCAT('''', COLUMN\_NAME, ''', ', COLUMN\_NAME)) FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_NAME = source\_feed\_name AND TABLE\_SCHEMA = DATABASE()),

'), ''In source only'' ',

'FROM ', source\_feed\_name, ' s ',

'LEFT JOIN ', target\_feed\_name, ' t ON ', @source\_col1, ' = ', @target\_col1, ' ',

'WHERE t.col\_1 IS NULL'

);

PREPARE stmt FROM @source\_only\_sql;

EXECUTE stmt;

DEALLOCATE PREPARE stmt;

SET @target\_only\_sql = CONCAT(

'INSERT INTO comparison\_results (source\_feed, target\_feed, record\_data, comparison\_status) ',

'SELECT ''', source\_feed\_name, ''', ''', target\_feed\_name, ''', ',

'JSON\_OBJECT(',

(SELECT GROUP\_CONCAT(CONCAT('''', COLUMN\_NAME, ''', ', COLUMN\_NAME)) FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_NAME = target\_feed\_name AND TABLE\_SCHEMA = DATABASE()),

'), ''In target only'' ',

'FROM ', target\_feed\_name, ' t ',

'LEFT JOIN ', source\_feed\_name, ' s ON ', @target\_col1, ' = ', @source\_col1, ' ',

'WHERE s.col\_1 IS NULL'

);

PREPARE stmt FROM @target\_only\_sql;

EXECUTE stmt;

DEALLOCATE PREPARE stmt;

SELECT CONCAT('Comparison between ', source\_feed\_name, ' and ', target\_feed\_name, ' is complete.') AS Status;

END$$

CREATE PROCEDURE run\_automated\_tests()

BEGIN

DECLARE test\_result VARCHAR(255);

DECLARE row\_count INT;

CALL generate\_feed('Feed1', 10, 10);

SELECT COUNT(\*) INTO row\_count FROM Feed1;

IF row\_count = 11 THEN

SELECT 'TC-01: Generate Feed-1 - PASSED' AS Test\_Result;

ELSE

SELECT 'TC-01: Generate Feed-1 - FAILED' AS Test\_Result;

END IF;

TRUNCATE TABLE duplicates;

CALL find\_and\_store\_duplicates('Feed1');

SELECT COUNT(\*) INTO row\_count FROM duplicates WHERE table\_name\_source = 'Feed1';

IF row\_count = 1 THEN

SELECT 'TC-04: Identify Duplicates in Feed-1 - PASSED' AS Test\_Result;

ELSE

SELECT 'TC-04: Identify Duplicates in Feed-1 - FAILED' AS Test\_Result;

END IF;

CALL replace\_duplicates\_with\_unique('Feed1');

SELECT COUNT(\*) INTO row\_count FROM Feed1;

IF row\_count = 10 THEN

SELECT 'TC-05: Replace Duplicates in Feed-1 - PASSED' AS Test\_Result;

ELSE

SELECT 'TC-05: Replace Duplicates in Feed-1 - FAILED' AS Test\_Result;

END IF;

CALL verify\_no\_duplicates('Feed1');

SELECT 'TC-06: Verify No Duplicates in Feed-1 - Check output manually' AS Test\_Result;

CALL generate\_feed('Feed2', 15, 15);

TRUNCATE TABLE comparison\_results;

CALL compare\_feeds('Feed2', 'Feed1');

SELECT COUNT(\*) INTO row\_count FROM comparison\_results;

IF row\_count > 0 THEN

SELECT 'TC-07: Compare Feed-2 to Feed-1 - PASSED (Results generated)' AS Test\_Result;

ELSE

SELECT 'TC-07: Compare Feed-2 to Feed-1 - FAILED (No results)' AS Test\_Result;

END IF;

END$$

DELIMITER ;

CALL run\_automated\_tests();

**. Entry & Exit Criteria**

* **Entry Criteria**: Database and SQL environment is set up, scripts are ready.
* **Exit Criteria**: All test cases executed, 100% pass rate on functional tests.

**Deliverables**

* Test Plan (this document)
* Test Cases execution results (screenshots + SQL outputs)
* Output files (duplicates, comparison output)
* Final Word Document as Project Report

Created & submitted by – neer soni