Perfect—here are only the steps (and SQL) needed to **load, de-duplicate, and keep a single no-duplicates table**. No extras. No synthetic links. No reclustering. PRS-latest-created wins when PRS is involved; otherwise newest created overall.

**1) Create the unified staging table**

CREATE TABLE STG\_USER (

stg\_id NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,

-- lineage

source\_system VARCHAR2(20) NOT NULL, -- PRS | MYQUEST | ECPM

source\_pk VARCHAR2(100) NOT NULL,

-- person

first\_name VARCHAR2(100),

middle\_name VARCHAR2(50),

last\_name VARCHAR2(100),

dob DATE,

gender\_code VARCHAR2(10),

-- contact / address (some sources lack these)

email VARCHAR2(256),

phone\_digits VARCHAR2(20),

address\_line\_1 VARCHAR2(256),

city\_name VARCHAR2(100),

zip\_code VARCHAR2(50),

-- cross-ids / relations (deterministic)

prs\_profile\_id VARCHAR2(100), -- PRS.PATIENT\_PROFILES\_ID | MYQUEST.PRS\_PROFILE\_ID

patient\_id VARCHAR2(100), -- MYQUEST.PATIENT\_ID | DUES.PATIENT\_ID

empi VARCHAR2(100), -- PRS.EMPI\_MEMRECNO

eid VARCHAR2(100), -- MYQUEST.EID

patient\_eid VARCHAR2(100), -- DUES.patient\_eid

user\_id VARCHAR2(100), -- lineage only

-- recency

created\_at\_src TIMESTAMP, -- PRS.DATE\_CREATED | MYQUEST.DATE\_CREATED | DUES.date\_created

-- normalized helpers

first\_up VARCHAR2(100),

middle\_up VARCHAR2(50),

last\_up VARCHAR2(100),

zip\_up VARCHAR2(50),

email\_norm VARCHAR2(256),

phone\_last7 VARCHAR2(10)

);

**2) Load the three sources (normalize while loading)**

**2.1 MYQUEST.MAIN\_PATIENT\_PROFILES → STG**

INSERT INTO STG\_USER (

source\_system, source\_pk,

first\_name, last\_name, dob, gender\_code,

email, phone\_digits,

address\_line\_1, city\_name, zip\_code,

prs\_profile\_id, patient\_id, eid, user\_id,

created\_at\_src,

first\_up, last\_up, zip\_up, email\_norm, phone\_last7

)

SELECT

'MYQUEST', TO\_CHAR(m.PATIENT\_ID),

INITCAP(TRIM(m.PRS\_FIRST\_NAME)), INITCAP(TRIM(m.PRS\_LAST\_NAME)),

m.PRS\_DATE\_OF\_BIRTH, m.PRS\_GENDER,

CASE WHEN REGEXP\_LIKE(LOWER(m.PRS\_EMAIL\_ID),'^[A-Za-z0-9.\_%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,}$')

THEN LOWER(m.PRS\_EMAIL\_ID) END,

REGEXP\_REPLACE(NVL(m.PRS\_PHONE,m.PRS\_SECONDARY\_PHONE), '\D'),

TRIM(m.PRS\_ADDRESS\_1), TRIM(m.PRS\_CITY), TRIM(m.PRS\_ZIP),

m.PRS\_PROFILE\_ID, TO\_CHAR(m.PATIENT\_ID), TO\_CHAR(m.EID), m.USER\_ID,

CAST(m.DATE\_CREATED AS TIMESTAMP),

UPPER(TRIM(REGEXP\_REPLACE(m.PRS\_FIRST\_NAME,'\s+',' '))),

UPPER(TRIM(REGEXP\_REPLACE(m.PRS\_LAST\_NAME,'\s+',' '))),

UPPER(TRIM(m.PRS\_ZIP)),

CASE WHEN REGEXP\_LIKE(LOWER(m.PRS\_EMAIL\_ID),'^[A-Za-z0-9.\_%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,}$')

THEN LOWER(m.PRS\_EMAIL\_ID) END,

SUBSTR(REGEXP\_REPLACE(NVL(m.PRS\_PHONE,m.PRS\_SECONDARY\_PHONE), '\D'), -7)

FROM MAIN\_PATIENT\_PROFILES@MYQUEST\_DB m;

**2.2 PRS.PATIENT\_PROFILES → STG *(LOGIN\_ID is not email)***

INSERT INTO STG\_USER (

source\_system, source\_pk,

first\_name, middle\_name, last\_name, dob, gender\_code,

prs\_profile\_id, empi, user\_id,

created\_at\_src,

first\_up, middle\_up, last\_up

)

SELECT

'PRS', TO\_CHAR(p.PATIENT\_PROFILES\_ID),

INITCAP(TRIM(p.FIRST\_NAME)), INITCAP(TRIM(p.MIDDLE\_INIT)), INITCAP(TRIM(p.LAST\_NAME)),

p.DATE\_OF\_BIRTH, p.GENDER\_CODE,

TO\_CHAR(p.PATIENT\_PROFILES\_ID), p.EMPI\_MEMRECNO, p.LOGIN\_ID,

CAST(p.DATE\_CREATED AS TIMESTAMP),

UPPER(TRIM(REGEXP\_REPLACE(p.FIRST\_NAME,'\s+',' '))),

UPPER(TRIM(REGEXP\_REPLACE(p.MIDDLE\_INIT,'\s+',' '))),

UPPER(TRIM(REGEXP\_REPLACE(p.LAST\_NAME,'\s+',' ')))

FROM PATIENT\_PROFILES@PRS\_DB p;

**2.3 ECPM.DUES\_PERSONS → STG**

INSERT INTO STG\_USER (

source\_system, source\_pk,

first\_name, middle\_name, last\_name, dob, gender\_code,

address\_line\_1, city\_name, zip\_code,

patient\_id, patient\_eid,

created\_at\_src,

first\_up, middle\_up, last\_up, zip\_up

)

SELECT

'ECPM', TO\_CHAR(d.person\_id),

INITCAP(TRIM(d.first\_name)), INITCAP(TRIM(d.middle\_name)), INITCAP(TRIM(d.last\_name)),

CAST(d.dob AS DATE), d.gender\_code,

TRIM(d.address\_line\_1), TRIM(d.city\_name), TRIM(d.zip\_code),

TO\_CHAR(d.patient\_id), TO\_CHAR(d.patient\_eid),

CAST(d.date\_created AS TIMESTAMP),

UPPER(TRIM(REGEXP\_REPLACE(d.first\_name,'\s+',' '))),

UPPER(TRIM(REGEXP\_REPLACE(d.middle\_name,'\s+',' '))),

UPPER(TRIM(REGEXP\_REPLACE(d.last\_name,'\s+',' '))),

UPPER(TRIM(d.zip\_code))

FROM DUES\_PERSONS@ECPM\_DB d;

**3) Create matching keys (deterministic + blocking)**

CREATE TABLE STG\_USER\_KEYS (

stg\_id NUMBER NOT NULL,

key\_type VARCHAR2(30) NOT NULL, -- 'PRS\_PROFILE\_ID','PATIENT\_ID','EMAIL','PHONE7','NAMEZIP','STREETNAME'

key\_val VARCHAR2(300) NOT NULL

);

-- Deterministic relations (always safe)

INSERT INTO STG\_USER\_KEYS SELECT stg\_id,'PRS\_PROFILE\_ID',prs\_profile\_id FROM STG\_USER WHERE prs\_profile\_id IS NOT NULL;

INSERT INTO STG\_USER\_KEYS SELECT stg\_id,'PATIENT\_ID', patient\_id FROM STG\_USER WHERE patient\_id IS NOT NULL;

-- Strong exacts / blocks

INSERT INTO STG\_USER\_KEYS SELECT stg\_id,'EMAIL', email\_norm FROM STG\_USER WHERE email\_norm IS NOT NULL;

INSERT INTO STG\_USER\_KEYS SELECT stg\_id,'PHONE7', phone\_last7 FROM STG\_USER WHERE phone\_last7 IS NOT NULL;

-- Tight blocks (keep user sets small)

INSERT INTO STG\_USER\_KEYS

SELECT stg\_id,'NAMEZIP',

SUBSTR(last\_up,1,1)||'|'||SUBSTR(first\_up,1,1)||'|'||zip\_up

FROM STG\_USER

WHERE last\_up IS NOT NULL AND first\_up IS NOT NULL AND zip\_up IS NOT NULL;

INSERT INTO STG\_USER\_KEYS

SELECT stg\_id,'STREETNAME',

NVL(REGEXP\_SUBSTR(address\_line\_1,'^\d+'),'')||'|'||

NVL(REGEXP\_SUBSTR(UPPER(address\_line\_1),'[A-Z]+'),'')||'|'||

last\_up

FROM STG\_USER

WHERE address\_line\_1 IS NOT NULL AND last\_up IS NOT NULL;

**4) Build user pairs (only within blocks)**

BEGIN EXECUTE IMMEDIATE 'DROP TABLE STG\_MATCH\_USERS PURGE'; EXCEPTION WHEN OTHERS THEN NULL; END;

/

CREATE TABLE STG\_MATCH\_USERS AS

SELECT DISTINCT

LEAST(k1.stg\_id, k2.stg\_id) AS stg\_a,

GREATEST(k1.stg\_id, k2.stg\_id) AS stg\_b,

k1.key\_type AS block\_type

FROM STG\_USER\_KEYS k1

JOIN STG\_USER\_KEYS k2

ON k1.key\_type = k2.key\_type

AND k1.key\_val = k2.key\_val

WHERE k1.stg\_id < k2.stg\_id

AND k1.key\_type IN ('PRS\_PROFILE\_ID','PATIENT\_ID','EMAIL','PHONE7','NAMEZIP','STREETNAME');

-- Optional guard: drop pairs with conflicting gender when both present

DELETE FROM STG\_MATCH\_USERS c

WHERE EXISTS (

SELECT 1

FROM STG\_USER a, STG\_USER b

WHERE a.stg\_id = c.stg\_a AND b.stg\_id = c.stg\_b

AND a.gender\_code IS NOT NULL AND b.gender\_code IS NOT NULL

AND a.gender\_code <> b.gender\_code

);

COMMIT;

**5) Fuzzy score each pair (dynamic weights)**

BEGIN EXECUTE IMMEDIATE 'DROP TABLE STG\_MATCH\_SCORED PURGE'; EXCEPTION WHEN OTHERS THEN NULL; END;

/

CREATE TABLE STG\_MATCH\_SCORED AS

SELECT

c.stg\_a, c.stg\_b, c.block\_type,

/\* similarities 0..100 \*/

( UTL\_MATCH.JARO\_WINKLER\_SIMILARITY(a.first\_up, b.first\_up)

+ UTL\_MATCH.JARO\_WINKLER\_SIMILARITY(a.last\_up, b.last\_up) )/2 AS name\_sim,

CASE WHEN a.middle\_up IS NOT NULL AND b.middle\_up IS NOT NULL

THEN UTL\_MATCH.JARO\_WINKLER\_SIMILARITY(a.middle\_up, b.middle\_up)

ELSE 100 END AS mid\_sim,

CASE WHEN a.dob IS NOT NULL AND b.dob IS NOT NULL THEN

CASE WHEN a.dob = b.dob THEN 100

WHEN ABS(a.dob - b.dob) <= 1 THEN 85

ELSE 0 END

ELSE 0 END AS dob\_sim,

LEAST(100,

0.70\*UTL\_MATCH.EDIT\_DISTANCE\_SIMILARITY(NVL(a.address\_line\_1,''), NVL(b.address\_line\_1,''))

+ 0.15\*UTL\_MATCH.JARO\_WINKLER\_SIMILARITY(NVL(a.city\_name,''), NVL(b.city\_name,''))

+ 0.15\*CASE WHEN NVL(a.zip\_up,'') = NVL(b.zip\_up,'') AND a.zip\_up IS NOT NULL THEN 100 ELSE 0 END

) AS addr\_sim,

CASE WHEN a.email\_norm IS NOT NULL AND b.email\_norm IS NOT NULL THEN

CASE WHEN a.email\_norm = b.email\_norm THEN 100

WHEN SUBSTR(a.email\_norm,1,INSTR(a.email\_norm,'@')-1)

= SUBSTR(b.email\_norm,1,INSTR(b.email\_norm,'@')-1) THEN 70

ELSE 0 END

ELSE 0 END AS email\_sim,

CASE WHEN a.phone\_digits IS NOT NULL AND b.phone\_digits IS NOT NULL AND a.phone\_digits = b.phone\_digits THEN 100

WHEN a.phone\_last7 IS NOT NULL AND b.phone\_last7 IS NOT NULL AND a.phone\_last7 = b.phone\_last7 THEN 80

ELSE 0 END AS phone\_sim,

/\* dynamic weights (only count if feature exists on BOTH sides) \*/

/\* base: name .30, middle .10, dob .30, addr .15, email .10, phone .05 \*/

CASE WHEN a.first\_up IS NOT NULL AND b.first\_up IS NOT NULL AND a.last\_up IS NOT NULL AND b.last\_up IS NOT NULL THEN 0.30 ELSE 0 END AS w\_name,

CASE WHEN a.middle\_up IS NOT NULL AND b.middle\_up IS NOT NULL THEN 0.10 ELSE 0 END AS w\_mid,

CASE WHEN a.dob IS NOT NULL AND b.dob IS NOT NULL THEN 0.30 ELSE 0 END AS w\_dob,

CASE WHEN (a.address\_line\_1 IS NOT NULL OR a.city\_name IS NOT NULL OR a.zip\_up IS NOT NULL)

AND (b.address\_line\_1 IS NOT NULL OR b.city\_name IS NOT NULL OR b.zip\_up IS NOT NULL)

THEN 0.15 ELSE 0 END AS w\_addr,

CASE WHEN a.email\_norm IS NOT NULL AND b.email\_norm IS NOT NULL THEN 0.10 ELSE 0 END AS w\_email,

CASE WHEN (a.phone\_digits IS NOT NULL OR a.phone\_last7 IS NOT NULL)

AND (b.phone\_digits IS NOT NULL OR b.phone\_last7 IS NOT NULL)

THEN 0.05 ELSE 0 END AS w\_phone,

/\* final score 0..1 \*/

CASE

WHEN ( (CASE WHEN a.first\_up IS NOT NULL AND b.first\_up IS NOT NULL AND a.last\_up IS NOT NULL AND b.last\_up IS NOT NULL THEN 0.30 ELSE 0 END)

+ (CASE WHEN a.middle\_up IS NOT NULL AND b.middle\_up IS NOT NULL THEN 0.10 ELSE 0 END)

+ (CASE WHEN a.dob IS NOT NULL AND b.dob IS NOT NULL THEN 0.30 ELSE 0 END)

+ (CASE WHEN (a.address\_line\_1 IS NOT NULL OR a.city\_name IS NOT NULL OR a.zip\_up IS NOT NULL)

AND (b.address\_line\_1 IS NOT NULL OR b.city\_name IS NOT NULL OR b.zip\_up IS NOT NULL)

THEN 0.15 ELSE 0 END)

+ (CASE WHEN a.email\_norm IS NOT NULL AND b.email\_norm IS NOT NULL THEN 0.10 ELSE 0 END)

+ (CASE WHEN (a.phone\_digits IS NOT NULL OR a.phone\_last7 IS NOT NULL)

AND (b.phone\_digits IS NOT NULL OR b.phone\_last7 IS NOT NULL)

THEN 0.05 ELSE 0 END) ) > 0

THEN ROUND(

( 0.30\*( ( UTL\_MATCH.JARO\_WINKLER\_SIMILARITY(a.first\_up, b.first\_up)

+ UTL\_MATCH.JARO\_WINKLER\_SIMILARITY(a.last\_up, b.last\_up) )/2 )/100

+ 0.10\*( CASE WHEN a.middle\_up IS NOT NULL AND b.middle\_up IS NOT NULL

THEN UTL\_MATCH.JARO\_WINKLER\_SIMILARITY(a.middle\_up, b.middle\_up)

ELSE 100 END )/100

+ 0.30\*( CASE WHEN a.dob IS NOT NULL AND b.dob IS NOT NULL THEN

CASE WHEN a.dob = b.dob THEN 100

WHEN ABS(a.dob - b.dob) <= 1 THEN 85

ELSE 0 END

ELSE 0 END )/100

+ 0.15\*( LEAST(100,

0.70\*UTL\_MATCH.EDIT\_DISTANCE\_SIMILARITY(NVL(a.address\_line\_1,''), NVL(b.address\_line\_1,''))

+ 0.15\*UTL\_MATCH.JARO\_WINKLER\_SIMILARITY(NVL(a.city\_name,''), NVL(b.city\_name,''))

+ 0.15\*CASE WHEN NVL(a.zip\_up,'') = NVL(b.zip\_up,'') AND a.zip\_up IS NOT NULL THEN 100 ELSE 0 END

) )/100

+ 0.10\*( CASE WHEN a.email\_norm IS NOT NULL AND b.email\_norm IS NOT NULL THEN

CASE WHEN a.email\_norm = b.email\_norm THEN 100

WHEN SUBSTR(a.email\_norm,1,INSTR(a.email\_norm,'@')-1)

= SUBSTR(b.email\_norm,1,INSTR(b.email\_norm,'@')-1) THEN 70

ELSE 0 END

ELSE 0 END )/100

+ 0.05\*( CASE

WHEN a.phone\_digits IS NOT NULL AND b.phone\_digits IS NOT NULL

AND a.phone\_digits = b.phone\_digits THEN 100

WHEN a.phone\_last7 IS NOT NULL AND b.phone\_last7 IS NOT NULL

AND a.phone\_last7 = b.phone\_last7 THEN 80

ELSE 0 END )/100

, 4)

ELSE 0

END AS score

FROM STG\_MATCH\_USERS c

JOIN STG\_USER a ON a.stg\_id = c.stg\_a

JOIN STG\_USER b ON b.stg\_id = c.stg\_b

;

**6) Drop the loser in each strong pair (PRS-latest-created wins)**

BEGIN EXECUTE IMMEDIATE 'DROP TABLE DEDUP\_LOSERS PURGE'; EXCEPTION WHEN OTHERS THEN NULL; END;

/

CREATE TABLE DEDUP\_LOSERS AS

SELECT DISTINCT

CASE

/\* Deterministic blocks always dedup \*/

WHEN s.block\_type IN ('PRS\_PROFILE\_ID','PATIENT\_ID') THEN

CASE

WHEN a.source\_system = 'PRS' AND b.source\_system <> 'PRS' THEN b.stg\_id

WHEN b.source\_system = 'PRS' AND a.source\_system <> 'PRS' THEN a.stg\_id

WHEN a.source\_system = 'PRS' AND b.source\_system = 'PRS'

THEN CASE WHEN NVL(a.created\_at\_src, DATE '1900-01-01') >= NVL(b.created\_at\_src, DATE '1900-01-01')

THEN b.stg\_id ELSE a.stg\_id END

ELSE -- neither PRS

CASE

WHEN NVL(a.created\_at\_src, DATE '1900-01-01') > NVL(b.created\_at\_src, DATE '1900-01-01') THEN b.stg\_id

WHEN NVL(a.created\_at\_src, DATE '1900-01-01') < NVL(b.created\_at\_src, DATE '1900-01-01') THEN a.stg\_id

ELSE CASE WHEN a.stg\_id < b.stg\_id THEN a.stg\_id ELSE b.stg\_id END

END

END

/\* Fuzzy blocks: only dedup if score ≥ 0.90 \*/

WHEN s.score >= 0.90 THEN

CASE

WHEN a.source\_system = 'PRS' AND b.source\_system <> 'PRS' THEN b.stg\_id

WHEN b.source\_system = 'PRS' AND a.source\_system <> 'PRS' THEN a.stg\_id

WHEN a.source\_system = 'PRS' AND b.source\_system = 'PRS'

THEN CASE WHEN NVL(a.created\_at\_src, DATE '1900-01-01') >= NVL(b.created\_at\_src, DATE '1900-01-01')

THEN b.stg\_id ELSE a.stg\_id END

ELSE -- neither PRS

CASE

WHEN NVL(a.created\_at\_src, DATE '1900-01-01') > NVL(b.created\_at\_src, DATE '1900-01-01') THEN b.stg\_id

WHEN NVL(a.created\_at\_src, DATE '1900-01-01') < NVL(b.created\_at\_src, DATE '1900-01-01') THEN a.stg\_id

ELSE CASE WHEN a.stg\_id < b.stg\_id THEN a.stg\_id ELSE b.stg\_id END

END

END

END AS loser\_id

FROM STG\_MATCH\_SCORED s

JOIN STG\_USER a ON a.stg\_id = s.stg\_a

JOIN STG\_USER b ON b.stg\_id = s.stg\_b

WHERE s.block\_type IN ('PRS\_PROFILE\_ID','PATIENT\_ID') OR s.score >= 0.90;

**7) Keep only survivors (your no-duplicates dataset)**

CREATE OR REPLACE VIEW STG\_USER\_SOT AS

SELECT s.\*

FROM STG\_USER s

LEFT JOIN DEDUP\_LOSERS L

ON L.loser\_id = s.stg\_id

WHERE L.loser\_id IS NULL;

STG\_USER\_SOT is your maintained, **no-duplicates** table/view. If you want it as a table:

CREATE TABLE SOT\_PERSON AS SELECT \* FROM STG\_USER\_SOT;

That’s the whole path from **load → match keys → pair → fuzzy → drop losers → survivors**, stopping exactly at the **maintained, deduplicated records**.