The primary key of each relation is underlined. The value 1 for attribute GENDER in APPLICANT relation indicates ‘Male’ and the value 2 for attribute GENDER in APPLICANT relation means ‘Female’. The value 1 for attribute SPLRES (special reservation) if APPLICANT relation indicates ‘Applicable' and the value 2 for attribute SPLRES in APPLICANT relation means ‘Not Applicable’.
Examination and Interview are conducted by two different agencies. The attribute RND of RANDOM relation should be Unique.
Perform the following using any two relational database management system.
-> Populate the APPLICANT relation with 20 tuples. 12 tuples with no special reservation and 8 tuples with special reservation.
-> Write a function to populate the value of REST in RANDOM relation
-> Write a function to populate the value of TEST attribute in APPLICANT relation from the TEST relation.
-> Write a procedure to populate the values for INTERVIEW attribute in APPLICANT relation from the INTERVIEW relation.
-> Write a procedure to populate the values of RND attribute in APPLICANT relation from the RANDOM relation
-> Update the value of AGGR in APPLICANT relation (AGGR - 50% of Test Marks + 50 % of Interview Marks)
-> Write the procedure to generate rank
-> Write a procedure to generate special reservation rank

Note :
Special reservation rank should be generated for Applicants from whom special reservation is applicable
Tie Break AGGR first ,TEST second ,INTERVIEW third , DATE OF BIRTH fourth and finally RANDOM NUMBER.

1. Database Schema Definition

The required tables include:

1. **APPLICANT** - Stores applicant details, test and interview scores, random numbers, and ranking details.
2. **TEST** - Contains test scores of applicants.
3. **INTERVIEW** - Stores interview scores.
4. **RANDOM** - Stores a unique random number for each applicant.

Table Creation Queries

```
CREATE TABLE APPLICANT (
    APPNO NUMBER(6) PRIMARY KEY,
    NAME VARCHAR(30),
    DOB DATE,
    GENDER NUMBER(1) CHECK(GENDER IN (1,2)),
    SPLRES NUMBER(1) CHECK(SPLRES IN (1,2)),
    TEST NUMBER(3),
```

```
    INTERVIEW NUMBER(3),
    AGGR NUMBER(6,3),
    RND NUMBER(10),
    RANK NUMBER(5),
    SPLRANK NUMBER(5)
);

CREATE TABLE TEST (
    APPNO NUMBER(6) PRIMARY KEY,
    NAME VARCHAR(30),
    TEST NUMBER(3),
    FOREIGN KEY(APPNO) REFERENCES APPLICANT(APPNO)
);

CREATE TABLE INTERVIEW (
    APPNO NUMBER(6) PRIMARY KEY,
    NAME VARCHAR(30),
    INTERVIEW NUMBER(3),
    FOREIGN KEY(APPNO) REFERENCES APPLICANT(APPNO)
);

CREATE TABLE RANDOM (
    APPNO NUMBER(6) PRIMARY KEY,
    NAME VARCHAR(30),
    RND NUMBER(10),
    FOREIGN KEY(APPNO) REFERENCES APPLICANT(APPNO)
);
```

2. Populating APPLICANT Table

- Insert 20 tuples (applicants).
- 8 applicants have special reservations (SPLRES = 1), and 12 do not (SPLRES = 2).
-

Insertion Queries

```
INSERT INTO APPLICANT (APPNO, NAME, DOB, GENDER, SPLRES) VALUES
(200001, 'PETER', '2001-04-02', 1, 1);
INSERT INTO APPLICANT (APPNO, NAME, DOB, GENDER, SPLRES) VALUES
(200002, 'JOHN', '2001-03-02', 1, 1);
INSERT INTO APPLICANT (APPNO, NAME, DOB, GENDER, SPLRES) VALUES
(200003, 'RAM', '2002-04-02', 1, 1);
INSERT INTO APPLICANT (APPNO, NAME, DOB, GENDER, SPLRES) VALUES
(200004, 'ANITHA', '2015-03-02', 2, 2);
...
-- Continue for all 20 applicants
```

```
COMMIT;
```

3. Populating Other Tables

- The **TEST** and **INTERVIEW** tables should be populated based on the applicants.

Inserting Data into TEST Table

```
INSERT INTO TEST (APPNO, NAME) SELECT APPNO, NAME FROM APPLICANT;
```

Inserting Data into INTERVIEW Table

```
INSERT INTO INTERVIEW (APPNO, NAME) SELECT APPNO, NAME FROM APPLICANT;
```

Inserting Data into RANDOM Table

```
INSERT INTO RANDOM (APPNO, NAME) SELECT APPNO, NAME FROM APPLICANT;
```

4. Populating TEST Marks

We update the **TEST** table with marks.

```
UPDATE TEST SET TEST = 98 WHERE APPNO = 200001;
UPDATE TEST SET TEST = 90 WHERE APPNO = 200002;
UPDATE TEST SET TEST = 85 WHERE APPNO = 200003;
...
-- Update for all 20 applicants
```

5. Populating INTERVIEW Marks

```
UPDATE INTERVIEW SET INTERVIEW = 50 WHERE APPNO = 200001;
UPDATE INTERVIEW SET INTERVIEW = 65 WHERE APPNO = 200002;
UPDATE INTERVIEW SET INTERVIEW = 70 WHERE APPNO = 200003;
...
-- Update for all 20 applicants
```

6. Updating APPLICANT Table with TEST and INTERVIEW Marks

We write **a procedure** to update **TEST** scores in the APPLICANT table.

```
DECLARE
    CURSOR C IS SELECT APPNO, TEST FROM TEST;
BEGIN
    FOR I IN C LOOP
        UPDATE APPLICANT
        SET TEST = I.TEST
        WHERE APPNO = I.APPNO;
    END LOOP;
    COMMIT;
END;
```

```
/
```

Similarly, we update **INTERVIEW** scores in the APPLICANT table.

```
DECLARE
    CURSOR C IS SELECT APPNO, INTERVIEW FROM INTERVIEW;
BEGIN
    FOR I IN C LOOP
        UPDATE APPLICANT
        SET INTERVIEW = I.INTERVIEW
        WHERE APPNO = I.APPNO;
    END LOOP;
    COMMIT;
END;
/
```

7. Populating RANDOM Numbers

Each applicant should have a unique **RND** value.

```
UPDATE RANDOM
SET RND = TRUNC(DBMS_RANDOM.VALUE(1000000000, 9999999999));
```

Then, update **APPLICANT** table with **RND** values.

```
DECLARE
    CURSOR C IS SELECT APPNO, RND FROM RANDOM;
BEGIN
    FOR I IN C LOOP
        UPDATE APPLICANT
        SET RND = I.RND
        WHERE APPNO = I.APPNO;
    END LOOP;
    COMMIT;
END;
/
```

8. Calculating Aggregate (AGGR)

The **AGGR** score is calculated as:

```
 UPDATE APPLICANT
SET AGGR = (TEST * 0.5 + INTERVIEW * 0.5);
```

9. Ranking Applicants

Applicants are ranked based on:

1. **AGGR** (Higher is better)

2. **TEST** (Higher is better)
3. **INTERVIEW** (Higher is better)
4. **DOB** (Older applicants get higher preference)
5. **RND** (Higher is better in case of ties)

Procedure to Assign Rank

```
CREATE OR REPLACE PROCEDURE RANK_APPLICANTS AS
BEGIN
    UPDATE APPLICANT
    SET RANK = (
        SELECT R FROM (
            SELECT APPNO,
                   RANK() OVER (ORDER BY AGGR DESC, TEST DESC,
INTERVIEW DESC, DOB ASC, RND DESC) AS R
            FROM APPLICANT
        ) WHERE APPLICANT.APPNO = APPNO
    );
    COMMIT;
END;
/
```

Procedure to Assign Special Reservation Rank

```
CREATE OR REPLACE PROCEDURE SPECIAL_RESERVATION_RANK AS
BEGIN
    UPDATE APPLICANT
    SET SPLRANK = (
        SELECT R FROM (
            SELECT APPNO,
                   RANK() OVER (ORDER BY AGGR DESC, TEST DESC,
INTERVIEW DESC, DOB ASC, RND DESC) AS R
            FROM APPLICANT WHERE SPLRES = 1
        ) WHERE APPLICANT.APPNO = APPNO
    );
    COMMIT;
END;
/
```

10. Running the Procedures

```
EXEC RANK_APPLICANTS;
EXEC SPECIAL_RESERVATION_RANK;
```

11. Viewing Final Result

```
SELECT * FROM APPLICANT ORDER BY RANK;
 SELECT * FROM APPLICANT WHERE SPLRES = 1 ORDER BY SPLRANK;
```