Stored Procedures:

1. Availability

```
SET search_path TO railway_reservation_system;

CREATE OR REPLACE FUNCTION availability(inp_trainno CHARACTER VARYING(7), dat
DATE, coach_no chtype)
  RETURNS INTEGER AS $BODY$
DECLARE
  rec         train%ROWTYPE;
  booked      INTEGER :=0;
  max_size    INTEGER :=0;
  MULTIPLIER  INTEGER :=0;
  VALIDITY_DAY BOOLEAN;
  DAY_OF_WEEK  INTEGER;
BEGIN

  SELECT EXTRACT(DOW FROM date dat)
  INTO DAY_OF_WEEK;

  IF DAY_OF_WEEK = 0
  THEN
    SELECT (FREQ).SUN
    INTO VALIDITY_DAY
    FROM TRAIN
    WHERE TRAINNO = INP_TRAINNO;
  END IF;
  IF DAY_OF_WEEK = 1
  THEN
    SELECT (FREQ).MON
    INTO VALIDITY_DAY
    FROM TRAIN
    WHERE TRAINNO = INP_TRAINNO;
  END IF;
  IF DAY_OF_WEEK = 2
  THEN
    SELECT (FREQ).TUE
    INTO VALIDITY_DAY
    FROM TRAIN
    WHERE TRAINNO = INP_TRAINNO;
  END IF;
  IF DAY_OF_WEEK = 3
```

```
  THEN
    SELECT (FREQ).WED
    INTO VALIDITY_DAY
    FROM TRAIN
    WHERE TRAINNO = INP_TRAINNO;
  END IF;
  IF DAY_OF_WEEK = 4
  THEN
    SELECT (FREQ).THU
    INTO VALIDITY_DAY
    FROM TRAIN
    WHERE TRAINNO = INP_TRAINNO;
  END IF;
  IF DAY_OF_WEEK = 5
  THEN
    SELECT (FREQ).FRI
    INTO VALIDITY_DAY
    FROM TRAIN
    WHERE TRAINNO = INP_TRAINNO;
  END IF;
  IF DAY_OF_WEEK = 6
  THEN
    SELECT (FREQ).SAT
    INTO VALIDITY_DAY
    FROM TRAIN
    WHERE TRAINNO = INP_TRAINNO;
  END IF;

  IF VALIDITY_DAY = FALSE
  THEN
    RAISE EXCEPTION 'TRAIN % DOESN"T RUN ON THE DATE YOU RESERVED THE
TICKET WITH PNR %', INP_TRAINNO, TICKET.PNR;
  END IF;

  SELECT capacity
  INTO max_size
  FROM coach
  WHERE coachcode = coach_no;

  SELECT *
  INTO rec
  FROM TRAIN
  WHERE TRAINNO = inp_trainno;
```

```
  IF COACH_NO = 'UR'
  THEN MULTIPLIER = rec.ur;
  END IF;
  IF COACH_NO = 'A2'
  THEN MULTIPLIER = rec.a2;
  END IF;
  IF COACH_NO = 'A3'
  THEN MULTIPLIER = rec.a3;
  END IF;
  IF COACH_NO = 'SL'
  THEN MULTIPLIER = rec.sl;
  END IF;
  IF COACH_NO = 'FC'
  THEN MULTIPLIER = rec.fc;
  END IF;
  IF COACH_NO = 'A1'
  THEN MULTIPLIER = rec.a1;
  END IF;
  IF COACH_NO = 'CC'
  THEN MULTIPLIER = rec.cc;
  END IF;
  IF COACH_NO = 'EX'
  THEN MULTIPLIER = rec.ex;
  END IF;

  max_size = max_size * MULTIPLIER;

  SELECT sum(total)
  INTO booked
  FROM tickets
  WHERE trainno = inp_trainno AND jdate = dat AND coach = coach_no AND (STATUS =
'CONFIRM' OR STATUS = 'WAITING');
  RAISE NOTICE '%', booked;
  IF booked IS NULL
  THEN RETURN max_size;
  ELSE RETURN max_size - booked;
  END IF;

END;

$BODY$ LANGUAGE plpgsql;
```

2. n days availability

```
CREATE OR REPLACE FUNCTION
railway_reservation_system.n_days_availability(inp_trainno CHARACTER VARYING, n
INTEGER,
                                                coach_no    railway_reservation_system.chtype)
  RETURNS SETOF holder AS
$BODY$
DECLARE
  --rec train%rowtype;
  booked     INTEGER;
  max_size   INTEGER := 0;
  available  INTEGER :=0;
  frequent   INTEGER :=0;
  day_part   CHARACTER VARYING(10);
  dat        DATE;
  i          INTEGER :=0;
  rec        holder%ROWTYPE;
  rec2       train%ROWTYPE;
  MULTIPLIER INTEGER :=0;
BEGIN

  SELECT capacity
  INTO max_size
  FROM coach
  WHERE coachcode = coach_no;

  SELECT *
  INTO rec2
  FROM TRAIN
  WHERE TRAINNO = inp_trainno;

  IF COACH_NO = 'UR'
  THEN MULTIPLIER = rec2.ur;
  END IF;
```

```
  IF COACH_NO = 'A2'
  THEN MULTIPLIER = rec2.a2;
  END IF;
  IF COACH_NO = 'A3'
  THEN MULTIPLIER = rec2.a3;
  END IF;
  IF COACH_NO = 'SL'
  THEN MULTIPLIER = rec2.sl;
  END IF;
  IF COACH_NO = 'FC'
  THEN MULTIPLIER = rec2.fc;
  END IF;
  IF COACH_NO = 'A1'
  THEN MULTIPLIER = rec2.a1;
  END IF;
  IF COACH_NO = 'CC'
  THEN MULTIPLIER = rec2.cc;
  END IF;
  IF COACH_NO = 'EX'
  THEN MULTIPLIER = rec2.ex;
  END IF;

  max_size = max_size * MULTIPLIER;

  FOR i IN 0..n - 1
  LOOP
    SELECT CURRENT_DATE + i
    INTO dat;
    SELECT sum(total)
    INTO booked
    FROM tickets
    WHERE trainno = inp_trainno AND jdate = dat AND coach = coach_no AND (STATUS =
'CONFIRM' OR STATUS = 'WAITING');
    available = max_size - booked;
    IF booked IS NULL
    THEN available = max_size;
    ELSE available = max_size - booked;
    END IF;
    SELECT
      inp_trainno,
      dat,
      available
    INTO rec;
```

```
   RETURN NEXT rec;
  END LOOP;
END;

$BODY$
LANGUAGE plpgsql;
```

3. trains between stations

```
CREATE OR REPLACE FUNCTION trains_between_stations(stn_code1 CHARACTER
VARYING, stn_code2 CHARACTER VARYING)
  RETURNS INTEGER AS $BODY$
DECLARE
  rec  train%ROWTYPE;
  rec2 trainstops%ROWTYPE;
  i    INTEGER :=0;
BEGIN
  FOR rec IN SELECT train.*
          FROM ((SELECT r1.trainno
                FROM trainstops AS r1
                  JOIN trainstops AS r2
                    ON r1.trainno = r2.trainno AND r1.stid = stn_code1 AND r2.stid = stn_code2
AND
                      r1.distance < r2.distance) AS r NATURAL JOIN train)
  LOOP
    RETURN NEXT rec;
  END LOOP;
END;

$BODY$ LANGUAGE plpgsql;
```

4. Fare Calculation

```
CREATE OR REPLACE FUNCTION fare_calculate(pnr_inp BPCHAR)
  RETURNS ticket_fare AS $BODY$
DECLARE
  rec  RECORD;
  rec2 RECORD;
  rec3 ticket_fare;
```

```
 dist numeric(7,2) :=0;
 dist1 numeric(7,2):=0;
 dist2 numeric(7,2):=0;
 cal_fare numeric(9,2) :=0;
 flag integer :=0;

BEGIN

 SELECT trainno,sourcest,destinationst,coach,total INTO rec FROM tickets
 WHERE
  pnr = pnr_inp;

 SELECT distance INTO dist1 from trainstops
 WHERE
  trainno = rec.trainno AND stid = rec.sourcest;

 raise notice '%', dist1;

 SELECT distance INTO dist2 from trainstops
 WHERE
  trainno = rec.trainno AND stid = rec.destinationst;

 raise notice '%' ,dist2;

 IF dist1 IS NULL
 THEN dist = dist2;
 ELSE
  dist = dist2-dist1;
 END IF;

 raise notice '%' ,dist;

 select bfare,mdistance into rec2 from basefare where coachcode = rec.coach;

 raise notice '%' ,rec2.bfare;
 raise notice '%' ,rec2.mdistance;

 IF dist <= rec2.mdistance THEN
  cal_fare = rec2.bfare*rec.total;
 ELSE
  SELECT 1 INTO flag from train WHERE trainno = rec.trainno AND traintype = 'SF';
  IF flag IS NULL THEN
   SELECT (fare*dist+reservationchrg)*rec.total INTO cal_fare from coach
```

```
    WHERE coachcode = rec.coach;
   ELSE
    SELECT (fare*dist+reservationchrg+superfastchrg)*rec.total INTO cal_fare from coach
    WHERE coachcode = rec.coach;
   END IF;
  END IF;

  raise notice '%' ,cal_fare;
  rec3.trainno = rec.trainno;
  rec3.source = rec.sourcest;
  rec3.destination = rec.destinationst;
  rec3.coach = rec.coach;
  rec3.total = rec.total;
  rec3.fare = cal_fare;

  RETURN rec3;
END;
$BODY$ LANGUAGE plpgsql;
```

5. Waiting or not waiting stamp (Booking trigger)

```
SET SEARCH_PATH TO RAILWAY_RESERVATION_SYSTEM;
DROP TRIGGER BOOKING_TRIGGER
ON TICKETS;
DROP FUNCTION INSTRIG_FUNC();
CREATE FUNCTION INSTRIG_FUNC()
  RETURNS TRIGGER AS
$$
DECLARE
  TOTAL_SEATS         INTEGER;
  TICKET              TICKETS%ROWTYPE;
  QUERY               VARCHAR(300);
  COACH_CAPICITY      INTEGER;
  BOOKED_SEATS        INTEGER;
  TOTAL_COACH         INTEGER;
  DAY_OF_WEEK         INTEGER;
  VALIDITY_DAY        BOOLEAN;
  JOURNEY_DATE        DATE;
  SOURCE_DISTANCE     DECIMAL(7, 2);
  DESTINATION_DISTANCE DECIMAL(7, 2);
```

```
BEGIN
  TOTAL_COACH := 0;
  TOTAL_SEATS := 0;
  COACH_CAPICITY := 0;
  BOOKED_SEATS := 0;

  FOR TICKET IN SELECT *
              FROM TICKETS
              WHERE STATUS IS NULL
  LOOP

    SELECT EXTRACT(DOW FROM date(TICKET.JDATE) :: DATE)
    INTO DAY_OF_WEEK;

    IF DAY_OF_WEEK = 0
    THEN
      SELECT (FREQ).SUN
      INTO VALIDITY_DAY
      FROM TRAIN
      WHERE TRAINNO = TICKET.TRAINNO;
    END IF;
    IF DAY_OF_WEEK = 1
    THEN
      SELECT (FREQ).MON
      INTO VALIDITY_DAY
      FROM TRAIN
      WHERE TRAINNO = TICKET.TRAINNO;
    END IF;
    IF DAY_OF_WEEK = 2
    THEN
      SELECT (FREQ).TUE
      INTO VALIDITY_DAY
      FROM TRAIN
      WHERE TRAINNO = TICKET.TRAINNO;
    END IF;
    IF DAY_OF_WEEK = 3
    THEN
      SELECT (FREQ).WED
      INTO VALIDITY_DAY
      FROM TRAIN
      WHERE TRAINNO = TICKET.TRAINNO;
    END IF;
    IF DAY_OF_WEEK = 4
```

```
    THEN
      SELECT (FREQ).THU
      INTO VALIDITY_DAY
      FROM TRAIN
      WHERE TRAINNO = TICKET.TRAINNO;
    END IF;
    IF DAY_OF_WEEK = 5
    THEN
      SELECT (FREQ).FRI
      INTO VALIDITY_DAY
      FROM TRAIN
      WHERE TRAINNO = TICKET.TRAINNO;
    END IF;
    IF DAY_OF_WEEK = 6
    THEN
      SELECT (FREQ).SAT
      INTO VALIDITY_DAY
      FROM TRAIN
      WHERE TRAINNO = TICKET.TRAINNO;
    END IF;

    IF VALIDITY_DAY = FALSE
    THEN
      RAISE EXCEPTION 'TRAIN % DOESN"T RUN ON THE DATE YOU RESERVED THE
TICKET WITH PNR %', TICKET.TRAINNO, TICKET.PNR;
    END IF;

    IF TICKET.COACH = '2S'
    THEN
      SELECT S2
      INTO TOTAL_COACH
      FROM TRAIN
      WHERE TRAINNO = TICKET.TRAINNO;
    END IF;
    IF TICKET.COACH = 'EX'
    THEN
      SELECT EX
      INTO TOTAL_COACH
      FROM TRAIN
      WHERE TRAINNO = TICKET.TRAINNO;
    END IF;
    IF TICKET.COACH = 'FC'
    THEN
```

```
  SELECT FC
  INTO TOTAL_COACH
  FROM TRAIN
  WHERE TRAINNO = TICKET.TRAINNO;
END IF;
IF TICKET.COACH = 'CC'
THEN
  SELECT CC
  INTO TOTAL_COACH
  FROM TRAIN
  WHERE TRAINNO = TICKET.TRAINNO;
END IF;
IF TICKET.COACH = 'A3'
THEN
  SELECT A3
  INTO TOTAL_COACH
  FROM TRAIN
  WHERE TRAINNO = TICKET.TRAINNO;
END IF;
IF TICKET.COACH = 'A2'
THEN
  SELECT A2
  INTO TOTAL_COACH
  FROM TRAIN
  WHERE TRAINNO = TICKET.TRAINNO;
END IF;
IF TICKET.COACH = 'A1'
THEN
  SELECT A1
  INTO TOTAL_COACH
  FROM TRAIN
  WHERE TRAINNO = TICKET.TRAINNO;
END IF;
IF TICKET.COACH = 'SL'
THEN
  SELECT SL
  INTO TOTAL_COACH
  FROM TRAIN
  WHERE TRAINNO = TICKET.TRAINNO;
END IF;

SELECT CAPACITY
INTO COACH_CAPICITY
```

```
   FROM COACH
   WHERE COACHCODE = TICKET.COACH;
   TOTAL_SEATS := TOTAL_COACH * COACH_CAPICITY;

   IF (TOTAL_SEATS <= 0)
   THEN
     RAISE EXCEPTION 'INVALID COACH TYPE % FOR TRAIN % WITH PNR %',
TICKET.COACH, TICKET.TRAINNO, TICKET.PNR;
   END IF;

   SELECT DISTANCE
   INTO SOURCE_DISTANCE
   FROM TRAINSTOPS
   WHERE TICKET.TRAINNO = TRAINSTOPS.TRAINNO AND TICKET.SOURCEST =
TRAINSTOPS.STID;
   SELECT DISTANCE
   INTO DESTINATION_DISTANCE
   FROM TRAINSTOPS
   WHERE TICKET.TRAINNO = TRAINSTOPS.TRAINNO AND TICKET.DESTINATIONST =
TRAINSTOPS.STID;

   SELECT SUM(TOTAL)
   INTO BOOKED_SEATS
   FROM (TICKETS
     JOIN TRAINSTOPS AS R1 ON TICKETS.TRAINNO = R1.TRAINNO AND
TICKETS.SOURCEST = R1.STID) JOIN TRAINSTOPS AS R2
       ON TICKETS.TRAINNO = R2.TRAINNO AND TICKETS.DESTINATIONST = R2.STID
   WHERE TICKETS.TRAINNO = TICKET.TRAINNO AND JDATE = TICKET.JDATE AND
COACH = TICKET.COACH AND STATUS IS NOT NULL AND
        STATUS != 'CANCEL' AND NOT (R2.DISTANCE <= SOURCE_DISTANCE OR
DESTINATION_DISTANCE <= R1.DISTANCE);

   IF (TOTAL_SEATS - BOOKED_SEATS < TICKET.TOTAL)
   THEN
     QUERY = 'UPDATE TICKETS SET STATUS = "WAITING" WHERE PNR = "' ||
TICKET.PNR || '"';
   ELSE
     QUERY = 'UPDATE TICKETS SET STATUS = "CONFIRM" WHERE PNR = "' ||
TICKET.PNR || '"';
   END IF;
   EXECUTE QUERY;
  END LOOP;
  RETURN NEW;
```

```
END;
$$
LANGUAGE 'PLPGSQL';
CREATE TRIGGER BOOKING_TRIGGER AFTER INSERT ON TICKETS FOR EACH
STATEMENT EXECUTE PROCEDURE INSTRIG_FUNC();
```

6. Cancel trigger

```
SET search_path TO railway_reservation_system;
DROP TRIGGER CANCEL_TRIGGER
ON TICKETS;
DROP FUNCTION CANCEL_TRIG_FUNC();
CREATE FUNCTION CANCEL_TRIG_FUNC()
  RETURNS TRIGGER AS
$$
DECLARE
  QUERY              VARCHAR(300);
  TOTAL_SEATS        INTEGER;
  COACH_CAPICITY     INTEGER;
  TOTAL_COACH        INTEGER;
  BOOKED_SEATS       INTEGER;
  TOTAL_AVLBL        INTEGER;
  TICKET             RECORD;
  NEWTICKET          RECORD;
  NEWPNR             VARCHAR(20);
  NEWCOACH           CHTYPE;
  NEWTRAINNO         VARCHAR(10);
  SOURCE_DISTANCE    DECIMAL(7, 2);
  DESTINATION_DISTANCE DECIMAL(7, 2);
BEGIN
  IF TG_OP = 'INSERT'
  THEN
    RETURN NEW;
  END IF;

  IF (TG_OP = 'UPDATE')
  THEN
    SELECT *
    INTO NEWTICKET
    FROM TICKETS
    WHERE STATUS = 'CAN';
```

```
   IF NEWTICKET.JDATE <= NOW()
   THEN
     RAISE EXCEPTION 'YOU CAN"T CANCEL THE TICKET WITH JOURNEY DATE LESS
THAN OR EQUAL TO TODAY"S DATE';
     RETURN NEW;
   END IF;

   NEWPNR := NEWTICKET.PNR;
   IF NEWPNR IS NULL
   THEN
     RETURN NEW;
   END IF;

   QUERY = 'UPDATE TICKETS SET STATUS = "CANCEL" WHERE PNR = '' ||
NEWTICKET.PNR || '';
   EXECUTE QUERY;

   NEWTRAINNO := NEWTICKET.TRAINNO;
   NEWCOACH := NEWTICKET.COACH;
   IF NEWTRAINNO IS NULL OR NEWCOACH IS NULL
   THEN
     RETURN NEW;
   END IF;

   IF NEWCOACH = 'UR'
   THEN
     SELECT UR
     INTO TOTAL_COACH
     FROM TRAIN
     WHERE TRAINNO = NEWTRAINNO;
   END IF;
   IF NEWCOACH = '2S'
   THEN
     SELECT S2
     INTO TOTAL_COACH
     FROM TRAIN
     WHERE TRAINNO = NEWTRAINNO;
   END IF;
   IF NEWCOACH = 'EX'
   THEN
     SELECT EX
     INTO TOTAL_COACH
```

```
  FROM TRAIN
  WHERE TRAINNO = NEWTRAINNO;
END IF;
IF NEWCOACH = 'FC'
THEN
  SELECT FC
  INTO TOTAL_COACH
  FROM TRAIN
  WHERE TRAINNO = NEWTRAINNO;
END IF;
IF NEWCOACH = 'CC'
THEN
  SELECT CC
  INTO TOTAL_COACH
  FROM TRAIN
  WHERE TRAINNO = NEWTRAINNO;
END IF;
IF NEWCOACH = 'A3'
THEN
  SELECT A3
  INTO TOTAL_COACH
  FROM TRAIN
  WHERE TRAINNO = NEWTRAINNO;
END IF;
IF NEWCOACH = 'A2'
THEN
  SELECT A2
  INTO TOTAL_COACH
  FROM TRAIN
  WHERE TRAINNO = NEWTRAINNO;
END IF;
IF NEWCOACH = 'A1'
THEN
  SELECT A1
  INTO TOTAL_COACH
  FROM TRAIN
  WHERE TRAINNO = NEWTRAINNO;
END IF;
IF NEWCOACH = 'SL'
THEN
  SELECT SL
  INTO TOTAL_COACH
  FROM TRAIN
```

```
    WHERE TRAINNO = NEWTRAINNO;
  END IF;

  SELECT CAPACITY
  INTO COACH_CAPICITY
  FROM COACH
  WHERE COACHCODE = NEWCOACH;
  TOTAL_SEATS := TOTAL_COACH * COACH_CAPICITY;

  << UPDATE_LOOP >>
  FOR TICKET IN SELECT *
            FROM TICKETS
            WHERE TRAINNO = NEWTRAINNO AND JDATE = NEWTICKET.JDATE AND
STATUS = 'WAITING' AND
                COACH = NEWCOACH :: CHTYPE
            ORDER BY TIME_STAMP
  LOOP

    SELECT DISTANCE
    INTO SOURCE_DISTANCE
    FROM TRAINSTOPS
    WHERE TICKET.TRAINNO = TRAINSTOPS.TRAINNO AND TICKET.SOURCEST =
TRAINSTOPS.STID;
    SELECT DISTANCE
    INTO DESTINATION_DISTANCE
    FROM TRAINSTOPS
    WHERE TICKET.TRAINNO = TRAINSTOPS.TRAINNO AND TICKET.DESTINATIONST =
TRAINSTOPS.STID;

    SELECT SUM(TOTAL)
    INTO BOOKED_SEATS
    FROM (TICKETS
      JOIN TRAINSTOPS AS R1 ON TICKETS.TRAINNO = R1.TRAINNO AND
TICKETS.SOURCEST = R1.STID) JOIN TRAINSTOPS AS R2
        ON TICKETS.TRAINNO = R2.TRAINNO AND TICKETS.DESTINATIONST = R2.STID
    WHERE
      TICKETS.TRAINNO = NEWTRAINNO AND JDATE = NEWTICKET.JDATE AND COACH
= NEWCOACH :: CHTYPE AND STATUS IS NOT NULL
      AND STATUS != 'CANCEL' AND STATUS != 'WAITING' AND
      NOT (R2.DISTANCE <= SOURCE_DISTANCE OR DESTINATION_DISTANCE <=
R1.DISTANCE);

    IF BOOKED_SEATS IS NULL
```

```
      THEN
        BOOKED_SEATS := 0;
      END IF;

      TOTAL_AVLBL := TOTAL_SEATS - BOOKED_SEATS;

      IF TICKET.TOTAL < TOTAL_AVLBL
      THEN
        QUERY = 'UPDATE TICKETS SET STATUS = "CONFIRM" WHERE PNR = "' ||
TICKET.PNR || '"';
        EXECUTE QUERY;
      END IF;
    END LOOP;
  END IF;
  RETURN NEW;
END;
$$
LANGUAGE 'PLPGSQL';
CREATE TRIGGER CANCEL_TRIGGER AFTER UPDATE OR INSERT ON TICKETS FOR
EACH STATEMENT EXECUTE PROCEDURE CANCEL_TRIG_FUNC();
```

SELECT QUERIES(SQL)

1.Retrieve details of all the trains that have source city DELHI (DLI) and destination city AHMEDABAD JN (ADI) which reach the destination on the same day of journey, in ascending order of total time taken.

```
SELECT train.*
FROM (SELECT
       r1.trainno,
       r2.arrtime - r1.deptime AS t
     FROM trainstops AS r1
       JOIN trainstops AS r2
         ON r1.trainno = r2.trainno AND r1.stid = 'DLI' AND r2.stid = 'ADI' AND r1.distance <
r2.distance AND
           r1.jday - r2.jday = 0) AS r3 NATURAL JOIN train
ORDER BY t;
```

2.List top 50 stations and trains in the terms of cleanliness as per passenger feedback

Top 50 trains:

```
SELECT TRAINNO FROM TRFEEDBACK ORDER BY CLEANLINESS
LIMIT 50;
```

Top 50 stations:

```
SELECT STID FROM STFEEDBACK ORDER BY CLEANLINESS
LIMIT 50;
```

3.List all trains between two given stations that travels via a particular route or suggest a combination of trains that form a chain between the source and destination.

```
SELECT
  R1.TRAINNO,
  R2.TRAINNO
FROM ((SELECT
         TRAINNO,
```

```
        DISTANCE AS D
       FROM TRAINSTOPS
       WHERE STID = 'ADI') as R3 NATURAL JOIN TRAINSTOPS) AS R1
  JOIN ((SELECT
          TRAINNO,
          DISTANCE AS E
        FROM TRAINSTOPS
        WHERE STID = 'BCT') as R4 NATURAL JOIN TRAINSTOPS) AS R2
    ON (R1.TRAINNO != R2.TRAINNO AND R1.STID = R2.STID AND R1.D < R1.DISTANCE
AND R2.E > R2.DISTANCE)
```

4. Show all trains whose departure from Ahmedabad (ADI) Station as source is scheduled after arrival of '19120/Somnath Ahmedabad Intercity Express' train at Ahmedabad (ADI) and destined to Delhi (DLI) on Saturday

```
SELECT DISTINCT trainno
FROM trains_between_stations('ADI', 'DLI') NATURAL JOIN trainstops AS X
WHERE X.deptime > (
  SELECT trainstops.arrtime
  FROM train
    NATURAL JOIN trainstops
  WHERE train.trainno = '19120' AND trainstops.stid = 'ADI'
)
```

5. List all Express Trains arriving at Lokmanya Tilak(LTT) Station whose cleanliness and punctuality is greater than or equal to the average cleanliness and punctuality of other trains whose feedback has been given.

```
SELECT TRAINNO
FROM
  trainstops
  NATURAL JOIN TRFEEDBACK
  NATURAL JOIN TRAIN
WHERE
  (STID = 'LTT' AND TRAINTYPE = 'EXP' AND
   CLEANLINESS > (
     SELECT AVG(CLEANLINESS)
     FROM TRFEEDBACK
       NATURAL JOIN TRAIN) AND
```

```
  PUNCTUALITY > (
    SELECT AVG(PUNCTUALITY)
    FROM TRFEEDBACK
      NATURAL JOIN TRAIN)
  );
```

7.Show top 10 express trains for which maximum number of reservations have been made, in First Class coach in the past 3 weeks.

```
SELECT
  trainno,
  SUM(total) AS RESERVATION
FROM TICKETS
  NATURAL JOIN TRAIN
WHERE reservationdate > (CURRENT_DATE - 21) AND reservationdate <= CURRENT_DATE
AND coach = 'FC' AND
      traintype = 'EXP'
GROUP BY trainno
ORDER BY RESERVATION DESC
LIMIT 10;
```

8. Find the total number of tickets booked this month

```
SELECT COUNT(pnr)
FROM (
  SELECT *
  FROM tickets
  WHERE reservationdate > current_date - 30 AND reservationdate <= CURRENT_DATE
) as t
```

9. Find the station with the maximum poitive feedback

```
SELECT *
FROM
  (
    SELECT
      (cleanliness + escalators + transportation + railfanning + safety + lodging) AS X,
      stid
    FROM stfeedback
    ORDER BY X
    LIMIT 1
  ) AS s
```

10. Find the train with the most negative feedback

```
SELECT *
FROM
  (
    SELECT
      (trfeedback.cleanliness + trfeedback.punctuality + trfeedback.tcktavlbl +
trfeedback.railfanning +
       trfeedback.safety) AS X,
      trainno
    FROM trfeedback
    ORDER BY X ASC
    LIMIT 1
  ) AS s
```

11. find the total number of administrator accounts in the database

```
SELECT count(userid)
FROM account
WHERE administrator = TRUE;
```

12. Find the number of tickets booked in the train with the most positive feedback for the next month

```
SELECT count(pnr) from tickets where trainno=(
  SELECT trainno
  FROM
    (
      SELECT
        (trfeedback.cleanliness + trfeedback.punctuality + trfeedback.tcktavlbl +
trfeedback.railfanning +
         trfeedback.safety) AS X,
        trainno
      FROM trfeedback
      ORDER BY X
      LIMIT 1
    ) AS s
) and jdate>current_date and jdate<=current_date+30;
```

13. Find the chain trains between 2 stations.

```
SELECT
  r5.trainno,
  r6.trainno,
  r5.stid
FROM (((SELECT
         trainno,
         distance AS d1
       FROM trainstops
       WHERE trainno NOT IN (SELECT r1.trainno
                             FROM trainstops AS r1
                               JOIN trainstops AS r2
                                 ON r1.trainno = r2.trainno AND r1.stid = 'SUNR' AND r2.stid = 'ADI') AND
           stid = 'SUNR') AS r4 NATURAL JOIN trainstops) AS r5
  JOIN ((SELECT
          r3.trainno,
          distance AS d2
        FROM trainstops AS r3
        WHERE r3.trainno NOT IN (SELECT r1.trainno
                                 FROM trainstops AS r1
                                   JOIN trainstops AS r2
                                     ON r1.trainno = r2.trainno AND r1.stid = 'SUNR' AND r2.stid = 'ADI')
AND
            stid = 'ADI') AS r4 NATURAL JOIN trainstops) AS r6 ON r5.trainno != r6.trainno AND
r5.stid = r6.stid AND
                                                      d1 < r5.distance AND d2 > r6.distance AND
                                                      (r6.deptime - r5.arrtime) <= INTERVAL '1 hour' AND
                                                      (r5.arrtime - r6.deptime) <= INTERVAL '1 hour');
```

Console Application:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
#include <libpq-fe.h>

static void exit_nicely(PGconn *conn)
{
    PQfinish(conn);
    exit(1);
}

void findResultOfQuery(PGresult *res)
{
    int rows = PQntuples(res);
    int columns = PQnfields(res);
    int i,j;
    for (i=0; i<columns; i++)
        printf("%-15s", PQfname(res, i));
    printf("\n");
    for(i=0; i<rows; i++)
    {
        for(j=0; j<columns; j++)
        {
            printf("%s ", PQgetvalue(res,i,j));
        }
        printf("\n");
    }
}

int main(int argc, char **argv)
{
    const char *conninfo;
    PGconn     *conn;
    PGresult   *res;
    const char *paramValues[1];
    int         paramLengths[1];
```

```
    int         paramFormats[1];
    unsigned long int    binaryIntVal;

    printf("\n\nTrying to establish a connection with server...\n\n");

    conn = PQconnectdb("hostaddr = '10.100.71.21' port = '5432' dbname = '201401058' user =
'201401058' password = '1234567890' connect_timeout = '10'");

    if (PQstatus(conn) != CONNECTION_OK)
    {
        fprintf(stderr, "Connection to database failed: %s",
                PQerrorMessage(conn));
        exit_nicely(conn);
    }

    printf("\n\nConnection is established successfully with server!!!\n\n");
    char str[2048];
    while(1){
        printf("\n\nEnter Your Query Here:\n\n");
        char pathstr[] = "set search_path to \"railway_reservation_system\";" ;
        gets(str);
        char * str2 = "end";
        if(strcmp(str,str2) == 0)
            break;
        strcat(pathstr, str);
        res = PQexec(conn,pathstr);

        if (PQresultStatus(res) != PGRES_TUPLES_OK)
        {
            fprintf(stderr, "SELECT failed: %s", PQerrorMessage(conn));
            PQclear(res);
            exit_nicely(conn);
        }
        else
        {
            fprintf(stderr, "Command is working\n");
            findResultOfQuery(res);
        }

        PQclear(res);
    }
    PQfinish(conn);
```

```
    return 0;
}
```