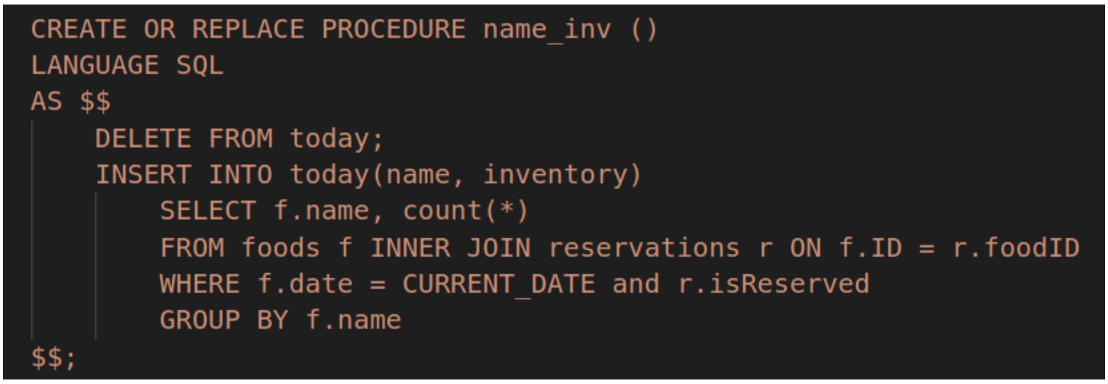
You can see output of project by run init.sh bash script in samad directory: https://github.com/alinaserinia6/Database.git

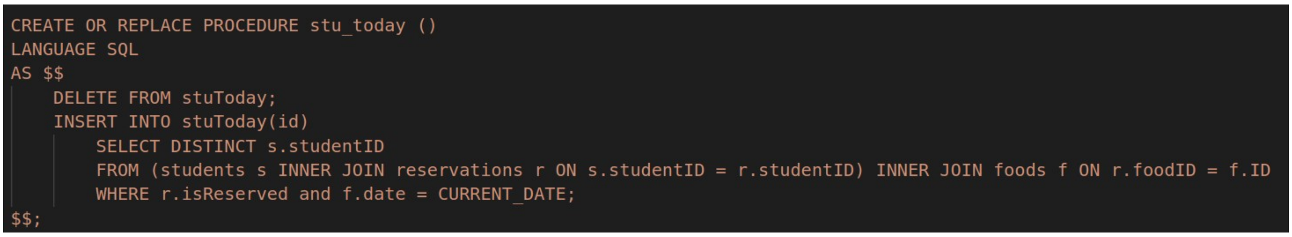
Procedures:

showToday: show both of name and number of foods that today reserved



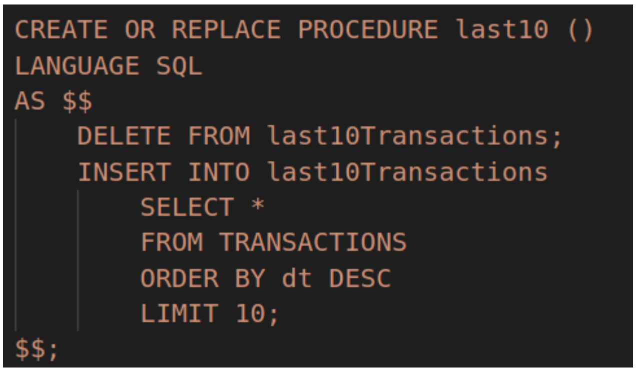
```
CREATE OR REPLACE PROCEDURE name_inv ()
LANGUAGE SQL
AS $$
    DELETE FROM today;
    INSERT INTO today(name, inventory)
        SELECT f.name, count(*)
        FROM foods f INNER JOIN reservations r ON f.ID = r.foodID
        WHERE f.date = CURRENT_DATE and r.isReserved
        GROUP BY f.name
$$;
```

showStuToday: show list of student that reserved food today



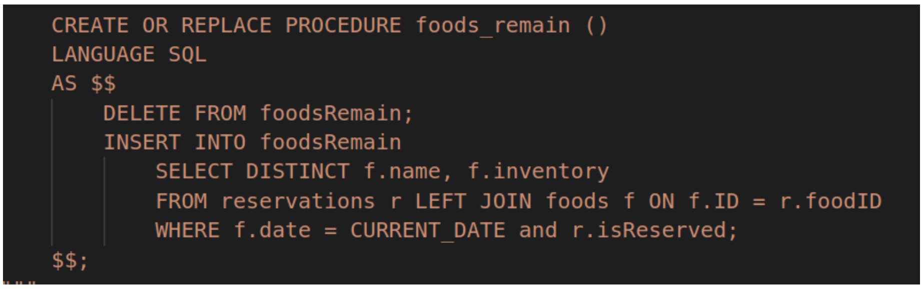
```
CREATE OR REPLACE PROCEDURE stu_today ()
LANGUAGE SQL
AS $$
    DELETE FROM stuToday;
    INSERT INTO stuToday(id)
        SELECT DISTINCT s.studentID
        FROM (students s INNER JOIN reservations r ON s.studentID = r.studentID) INNER JOIN foods f ON r.foodID = f.ID
        WHERE r.isReserved and f.date = CURRENT_DATE;
$$;
```

showLast10Transactions:



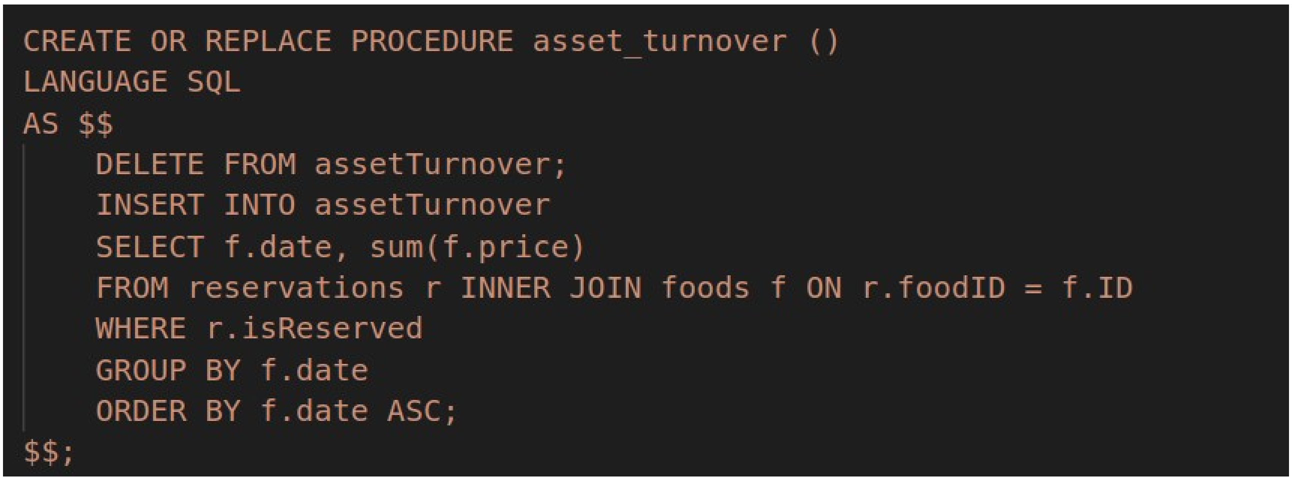
```
CREATE OR REPLACE PROCEDURE last10 ()
LANGUAGE SQL
AS $$
    DELETE FROM last10Transactions;
    INSERT INTO last10Transactions
        SELECT *
        FROM TRANSACTIONS
        ORDER BY dt DESC
        LIMIT 10;
$$;
```

show remian food for today



```
CREATE OR REPLACE PROCEDURE foods_remain ()
LANGUAGE SQL
AS $$
    DELETE FROM foodsRemain;
    INSERT INTO foodsRemain
        SELECT DISTINCT f.name, f.inventory
        FROM reservations r LEFT JOIN foods f ON f.ID = r.foodID
        WHERE f.date = CURRENT_DATE and r.isReserved;
$$;
```

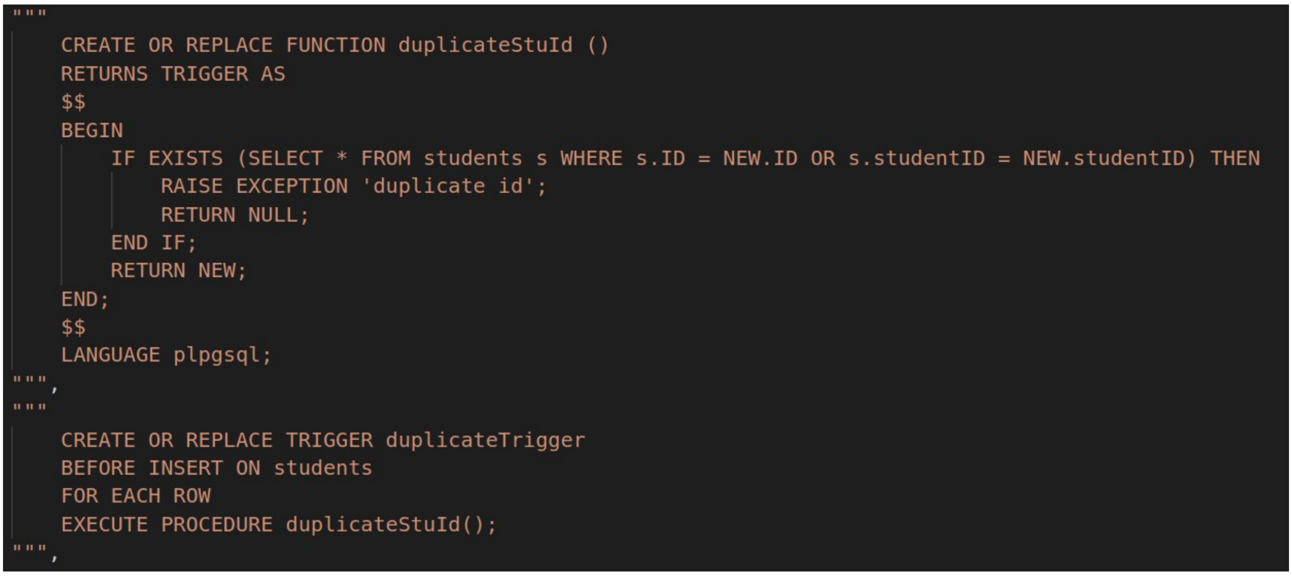
گردش مالی در روز



```
CREATE OR REPLACE PROCEDURE asset_turnover ()
LANGUAGE SQL
AS $$
    DELETE FROM assetTurnover;
    INSERT INTO assetTurnover
    SELECT f.date, sum(f.price)
    FROM reservations r INNER JOIN foods f ON r.foodID = f.ID
    WHERE r.isReserved
    GROUP BY f.date
    ORDER BY f.date ASC;
$$;
```

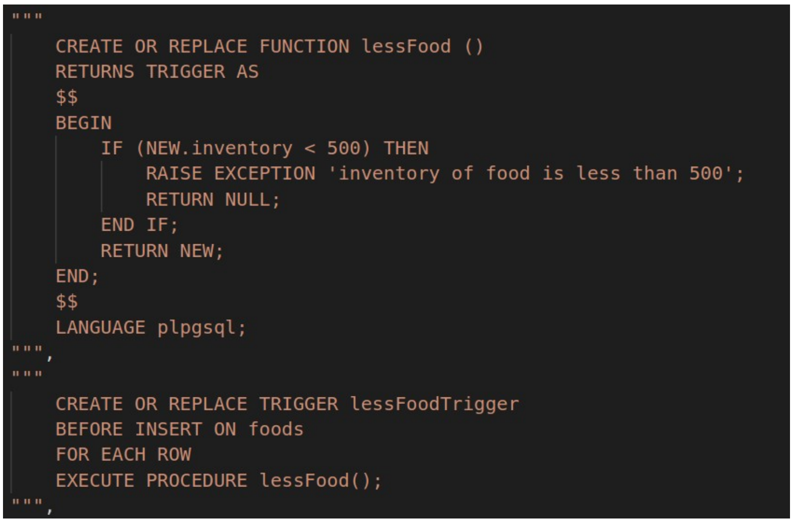
TRIGGERS:

register new user with same id or student id



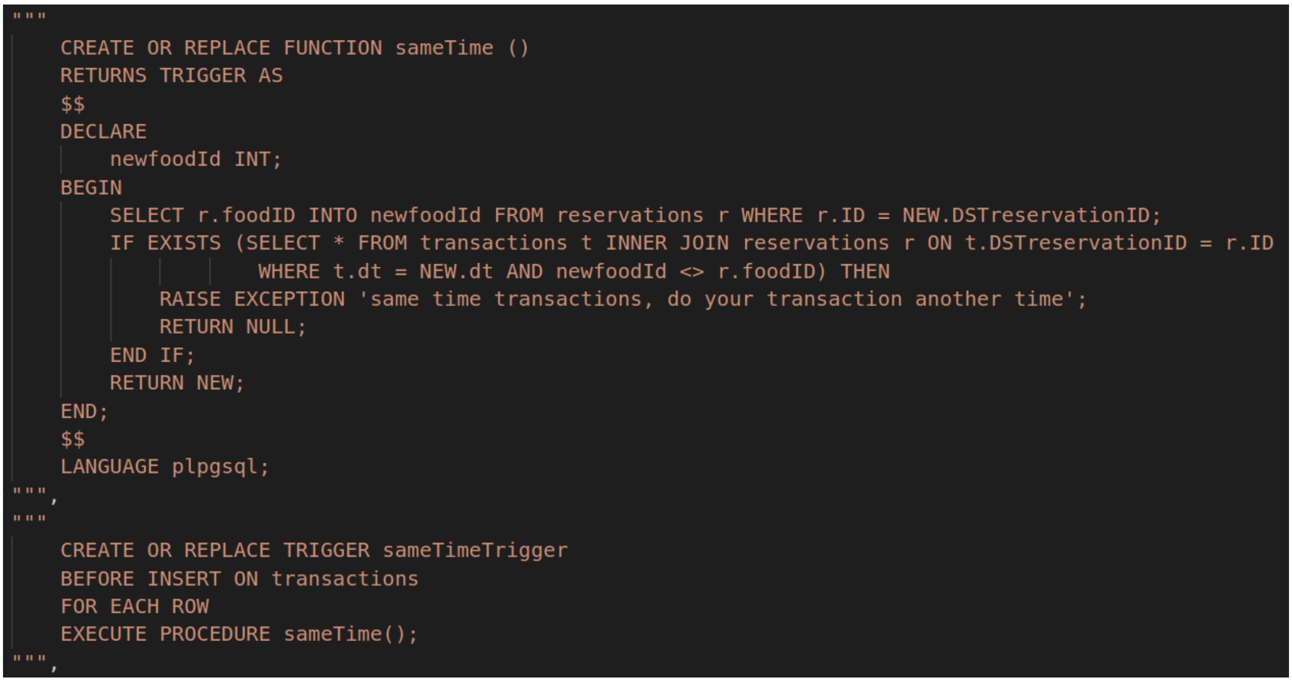
```
"""
    CREATE OR REPLACE FUNCTION duplicateStuId ()
    RETURNS TRIGGER AS
    $$
    BEGIN
        IF EXISTS (SELECT * FROM students s WHERE s.ID = NEW.ID OR s.studentID = NEW.studentID) THEN
            RAISE EXCEPTION 'duplicate id';
            RETURN NULL;
        END IF;
        RETURN NEW;
    END;
    $$
    LANGUAGE plpgsql;
""",
"""
    CREATE OR REPLACE TRIGGER duplicateTrigger
    BEFORE INSERT ON students
    FOR EACH ROW
    EXECUTE PROCEDURE duplicateStuId();
""",
```

every food inventory should be greater or equal to 500



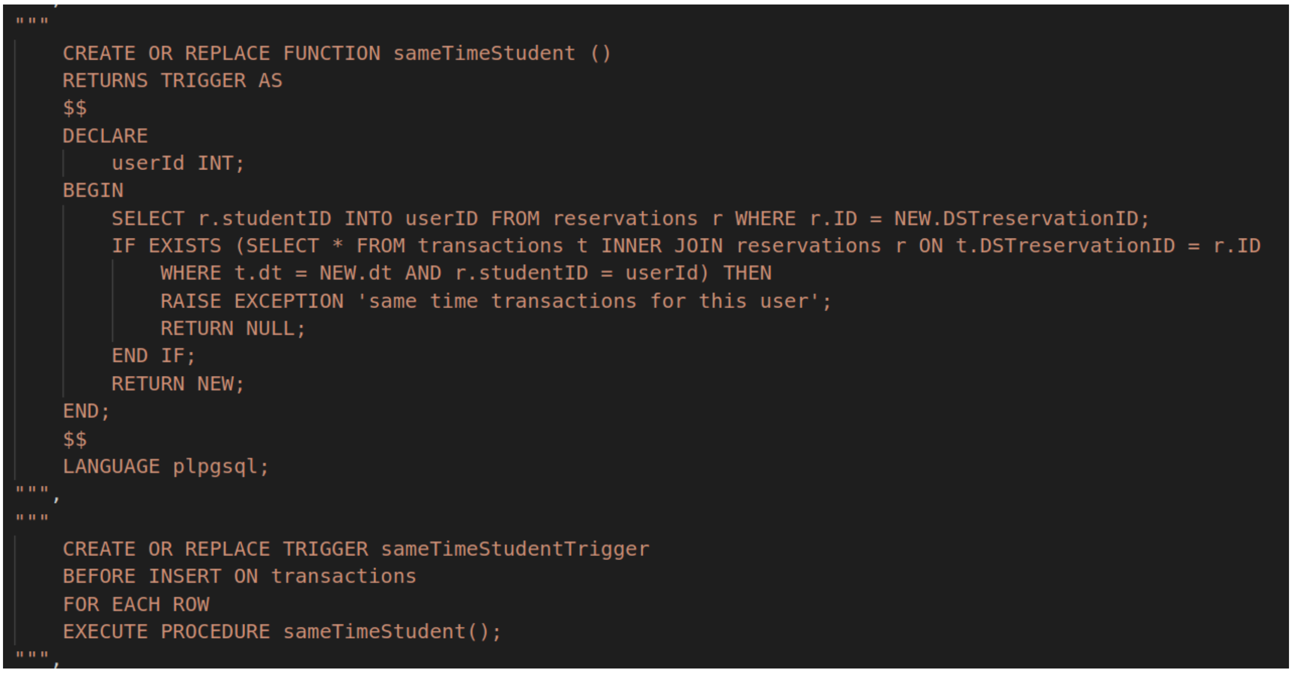
```
"""
    CREATE OR REPLACE FUNCTION lessFood ()
    RETURNS TRIGGER AS
    $$
    BEGIN
        IF (NEW.inventory < 500) THEN
            RAISE EXCEPTION 'inventory of food is less than 500';
            RETURN NULL;
        END IF;
        RETURN NEW;
    END;
    $$
    LANGUAGE plpgsql;
""",
"""
    CREATE OR REPLACE TRIGGER lessFoodTrigger
    BEFORE INSERT ON foods
    FOR EACH ROW
    EXECUTE PROCEDURE lessFood();
""",
```

reserve two food in same time



```
"""
    CREATE OR REPLACE FUNCTION sameTime ()
    RETURNS TRIGGER AS
    $$
    DECLARE
        newfoodId INT;
    BEGIN
        SELECT r.foodID INTO newfoodId FROM reservations r WHERE r.ID = NEW.DSTreservationID;
        IF EXISTS (SELECT * FROM transactions t INNER JOIN reservations r ON t.DSTreservationID = r.ID
                    WHERE t.dt = NEW.dt AND newfoodId <> r.foodID) THEN
            RAISE EXCEPTION 'same time transactions, do your transaction another time';
            RETURN NULL;
        END IF;
        RETURN NEW;
    END;
    $$
    LANGUAGE plpgsql;
""",
"""
    CREATE OR REPLACE TRIGGER sameTimeTrigger
    BEFORE INSERT ON transactions
    FOR EACH ROW
    EXECUTE PROCEDURE sameTime();
""",
```

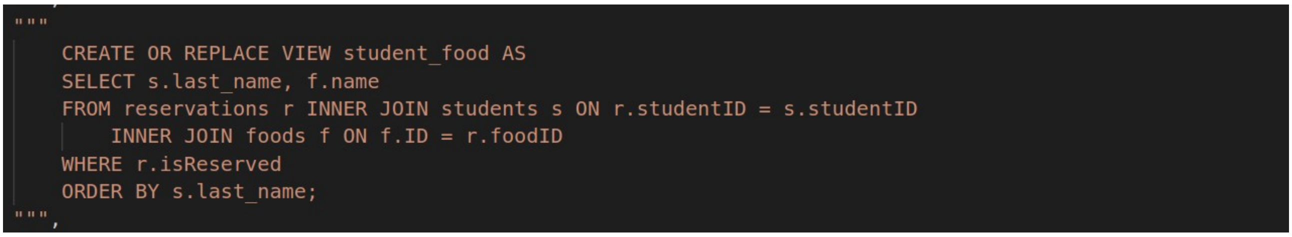
one user reserve at least 2 food in same time



```
"""
    CREATE OR REPLACE FUNCTION sameTimeStudent ()
    RETURNS TRIGGER AS
    $$
    DECLARE
        userId INT;
    BEGIN
        SELECT r.studentID INTO userID FROM reservations r WHERE r.ID = NEW.DSTreservationID;
        IF EXISTS (SELECT * FROM transactions t INNER JOIN reservations r ON t.DSTreservationID = r.ID
            WHERE t.dt = NEW.dt AND r.studentID = userId) THEN
            RAISE EXCEPTION 'same time transactions for this user';
            RETURN NULL;
        END IF;
        RETURN NEW;
    END;
    $$
    LANGUAGE plpgsql;
""",
"""
    CREATE OR REPLACE TRIGGER sameTimeStudentTrigger
    BEFORE INSERT ON transactions
    FOR EACH ROW
    EXECUTE PROCEDURE sameTimeStudent();
""",
```

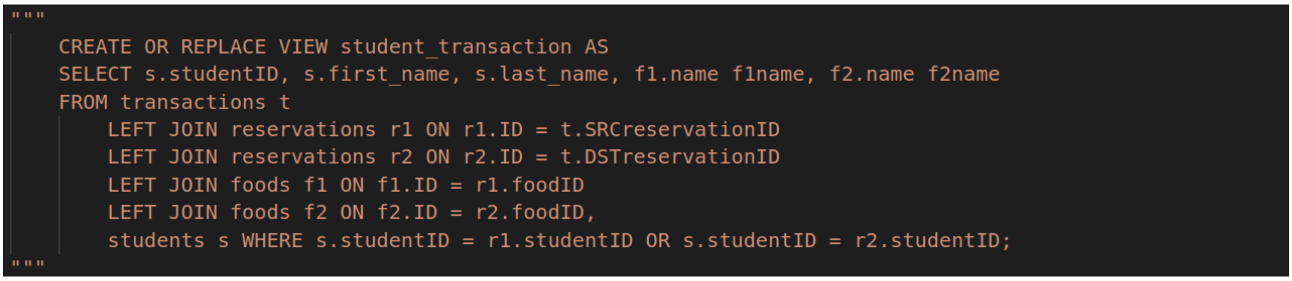
VIEWS:

student_food:



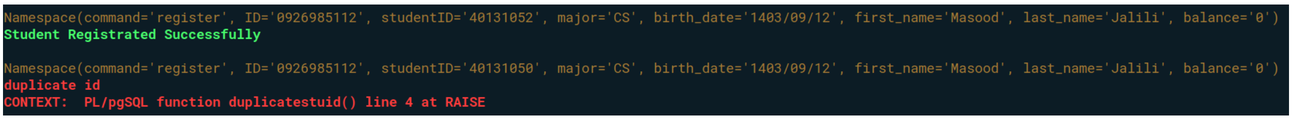
```
"""
    CREATE OR REPLACE VIEW student_food AS
    SELECT s.last_name, f.name
    FROM reservations r INNER JOIN students s ON r.studentID = s.studentID
        INNER JOIN foods f ON f.ID = r.foodID
    WHERE r.isReserved
    ORDER BY s.last_name;
""",
```

student_transactions:



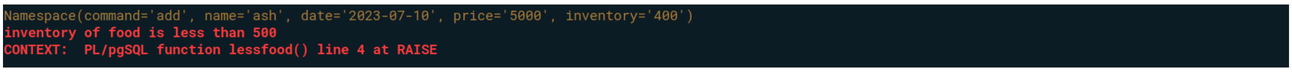
```
"""
    CREATE OR REPLACE VIEW student_transaction AS
    SELECT s.studentID, s.first_name, s.last_name, f1.name f1name, f2.name f2name
    FROM transactions t
        LEFT JOIN reservations r1 ON r1.ID = t.SRCreservationID
        LEFT JOIN reservations r2 ON r2.ID = t.DSTreservationID
        LEFT JOIN foods f1 ON f1.ID = r1.foodID
        LEFT JOIN foods f2 ON f2.ID = r2.foodID,
        students s WHERE s.studentID = r1.studentID OR s.studentID = r2.studentID;
"""
```

output of duplicate id trigger:



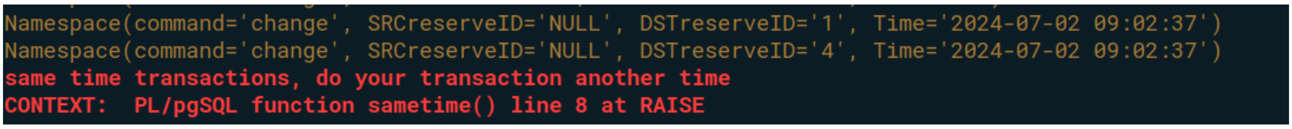
```
Namespace(command='register', ID='0926985112', studentID='40131052', major='CS', birth_date='1403/09/12', first_name='Masood', last_name='Jalili', balance='0')
Student Registrated Successfully

Namespace(command='register', ID='0926985112', studentID='40131050', major='CS', birth_date='1403/09/12', first_name='Masood', last_name='Jalili', balance='0')
duplicate id
CONTEXT:  PL/pgSQL function duplicatestuid() line 4 at RAISE
```

output of less than 500 food trigger:



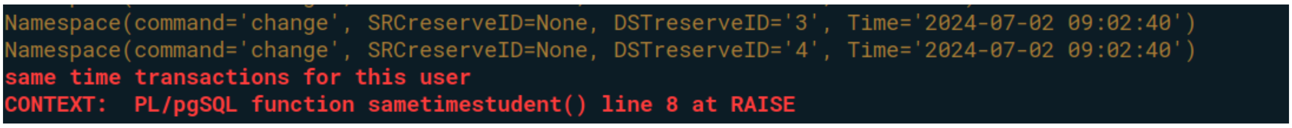
```
Namespace(command='add', name='ash', date='2023-07-10', price='5000', inventory='400')
inventory of food is less than 500
CONTEXT:  PL/pgSQL function lessfood() line 4 at RAISE
```

output of same time trigger:



```
Namespace(command='change', SRCreserveID='NULL', DSTreserveID='1', Time='2024-07-02 09:02:37')
Namespace(command='change', SRCreserveID='NULL', DSTreserveID='4', Time='2024-07-02 09:02:37')
same time transactions, do your transaction another time
CONTEXT:  PL/pgSQL function sametime() line 8 at RAISE
```

output of same time by user trigger:



```
Namespace(command='change', SRCreserveID=None, DSTreserveID='3', Time='2024-07-02 09:02:40')
Namespace(command='change', SRCreserveID=None, DSTreserveID='4', Time='2024-07-02 09:02:40')
same time transactions for this user
CONTEXT:  PL/pgSQL function sametimestudent() line 8 at RAISE
```