```
-- ADVANCED DATABASE CONCEPTS: ASSIGNMENT 1
-- (Please find Query 1 at the last due to schema changes)


-- Query 2

SELECT DISTINCT e1.deptName, e1.salary
FROM employedBy e1
WHERE e1.salary >= ALL (
    SELECT e2.salary
    FROM employedBy e2
    WHERE e1.deptName = e2.deptName
);


-- Query 3

SELECT DISTINCT s1.sid, s1.sname
FROM Student s1, employedBy e1
WHERE s1.sid = e1.sid
  AND EXISTS (
      SELECT 1
      FROM hasFriend f, Student s2, employedBy e2
      WHERE f.sid1 = s1.sid
        AND f.sid2 = s2.sid
        AND s2.sid = e2.sid
        AND e1.deptName = e2.deptName
        AND e1.salary > e2.salary
  )
  AND NOT EXISTS (
      SELECT 1
      FROM hasFriend f, Student s2, employedBy e2
      WHERE f.sid1 = s1.sid
        AND f.sid2 = s2.sid
        AND s2.sid = e2.sid
        AND e1.deptName = e2.deptName
        AND e1.salary <= e2.salary
  );


-- Query 4

SELECT DISTINCT e.deptName
FROM employedBy e
WHERE EXISTS (
    SELECT *
    FROM Student s
    WHERE s.sid = e.sid
      AND s.homeCity <> 'Indianapolis'
);


-- Query 5

SELECT DISTINCT s.sid, s.sname
FROM Student s, employedBy e
WHERE s.sid = e.sid
  AND s.homeCity = 'Bloomington'
  AND e.salary > 20000
```

```
  AND EXISTS (
    SELECT 1
    FROM hasFriend f
    WHERE f.sid1 = s.sid
  );


-- Query 6

SELECT DISTINCT d1.deptName, d2.deptName
FROM Department d1, Department d2
WHERE d1.mainOffice = d2.mainOffice
  AND d1.deptName <> d2.deptName;


-- Query 7

SELECT DISTINCT s1.sid, s1.sname
FROM Student s1
WHERE NOT EXISTS (
  SELECT 1
  FROM Student s2
  WHERE EXISTS (
    SELECT 1
    FROM hasFriend h
    WHERE h.sid1 = s1.sid
    AND h.sid2 = s2.sid
  )
  AND s1.homeCity = s2.homeCity
);


-- Query 8

SELECT DISTINCT m.major
FROM Major m
WHERE NOT EXISTS (
    SELECT 1
    FROM studentMajor sm1, studentMajor sm2
    WHERE sm1.major = m.major
      AND sm2.major = m.major
      AND sm1.sid != sm2.sid
      AND EXISTS (
          SELECT 1
          FROM studentMajor sm3
          WHERE sm3.major = m.major
            AND sm3.sid != sm1.sid
            AND sm3.sid != sm2.sid
      )
);


-- Query 9

SELECT DISTINCT s1.sid, s1.sname, e.salary
FROM Student s1, employedBy e
WHERE e.sid = s1.sid
AND EXISTS (
    SELECT 1
```

```
    FROM hasFriend f1, hasFriend f2, studentMajor sm1, studentMajor sm2
    WHERE f1.sid1 = s1.sid
      AND f1.sid2 = sm1.sid
      AND sm1.major != 'Mathematics'
      AND f2.sid1 = s1.sid
      AND f2.sid2 = sm2.sid
      AND sm1.major = sm2.major
      AND f1.sid2 != f2.sid2
);


-- Query 10

SELECT s1.sid, s1.sname
FROM Student s1
WHERE EXISTS (
    SELECT 1
    FROM Department d, employedBy w
    WHERE d.deptName = w.deptName
    AND s1.sid = w.sid
    AND d.mainOffice = 'LuddyHall'
    AND EXISTS(
        SELECT 1
        FROM hasFriend f
        WHERE f.sid1 = s1.sid
        AND f.sid2 IN (
            SELECT s2.sid
            FROM Student s2
            WHERE s2.homeCity != 'Bloomington'
        )
    )
);


-- Query 11

SELECT DISTINCT s1.sid
FROM Student s1
WHERE NOT EXISTS (
    SELECT 1
    FROM Student s2, hasFriend f
    WHERE f.sid1 = s1.sid AND f.sid2 = s2.sid
    AND NOT EXISTS (
        SELECT 1
        FROM studentMajor sm1, studentMajor sm2
        WHERE sm1.sid = s1.sid AND sm2.sid = s2.sid
        AND sm1.major = sm2.major
        AND sm1.sid != sm2.sid
    )
);


-- Query 12

SELECT DISTINCT s1.sid, s2.sid
FROM Student s1, Student s2
WHERE s1.sid <> s2.sid
AND NOT EXISTS (
    SELECT 1
```

```
    FROM hasFriend f1
    WHERE f1.sid1 = s1.sid
    AND NOT EXISTS (
        SELECT 1
        FROM hasFriend f2
        WHERE f2.sid1 = s2.sid
        AND f1.sid2 = f2.sid2
    )
);


-- Query 13

SELECT m.major
FROM Major m
WHERE NOT EXISTS (
    SELECT 1
    FROM Student s, studentMajor sm
    WHERE s.sid = sm.sid
    AND sm.major = m.major
    AND s.homeCity = 'Bloomington'
);


-- Query 18 (b)
SELECT TRUE = SOME (
    SELECT TRUE = ALL (
        SELECT s1.sid = s2.sid
        FROM Student s1, Student s2
        WHERE (s1.sid, m.major) IN (SELECT * FROM studentMajor)
        AND (s2.sid, m.major) IN (SELECT * FROM studentMajor)
    )
    FROM Major m
);


-- Query 19 (b)
SELECT NOT EXISTS (
    SELECT 1
    FROM Student s
    WHERE NOT (
        EXISTS (
            SELECT 1
            FROM employedBy w
            WHERE w.sid = s.sid
        )
        AND EXISTS (
            SELECT 1
            FROM studentMajor sm1, studentMajor sm2
            WHERE sm1.sid = s.sid
            AND sm2.sid = s.sid
            AND sm1.major <> sm2.major
        )
    )
);


-- Query 20 (b)
SELECT NOT EXISTS (
```

```
    SELECT 1
    FROM hasFriend f, employedBy wf1, employedBy wf2
    WHERE f.sid1 = wf1.sid
      AND f.sid2 = wf2.sid
      AND wf1.deptName <> wf2.deptName
);


-- Query 1

-- Example 1: Inserting Data Without Foreign Key Violations
INSERT INTO studentMajor VALUES (1001, 'Biology');
-- Since sid 1001 exists in the Student table, and 'Biology' exists in the Major
table,
-- this insertion works perfectly because both the foreign key constraints on sid
and major are satisfied.

-- Example 2: Inserting Data with Foreign Key Violations
INSERT INTO employedBy VALUES (1040, 'CS', 60000);
-- This insertion fails because there is no sid = 1040 in the Student table.
-- The foreign key constraint on employedBy(sid) references Student(sid),
-- so inserting a record without a matching student violates this constraint and
generates an error

-- Example 3: Inserting Duplicate Primary Key
INSERT INTO Department VALUES ('CS', 'LuddyHall');
-- The deptName column is a primary key in the Department table,
-- meaning all department names must be unique. Since 'CS' already exists in the
Department table,
-- the insertion fails due to a primary key violation.

-- Example 4: Altering Primary Key Constraint and Inserting Duplicate
ALTER TABLE Student DROP CONSTRAINT student_pkey CASCADE;
INSERT INTO Student VALUES (1001, 'Jordan', 'Chicago');
-- By dropping the primary key constraint on sid,
-- it is now possible to insert a duplicate sid into the Student table.
-- This insertion succeeds because there is no primary key constraint to enforce
uniqueness.
```