Jai Malhi
301457742

CMPT354 - Assignment 3

Part 1: SQL Programming

1. Create the above database schema using CREATE TABLE statements, including primary key constraints, and the constraint that salary is integer in the range [5000, 20000]. You can assume CHAR(20) type for all other attributes.

- CREATE TABLE Employees (
  eid CHAR(20),
  name CHAR(20),
  dept CHAR(20),
  salary INTEGER,
  PRIMARY KEY (eid),
  CHECK (salary >= 5000 AND salary <= 20000)
  );
- CREATE TABLE Sales (
  dept CHAR(20),
  item CHAR(20),
  PRIMARY KEY (dept, item)
  );
- CREATE TABLE Types (
  item CHAR(20),
  color CHAR(20),
  PRIMARY KEY (item, color)
  );

```
cmpt354_a3=> \dt
              List of relations
 Schema |   Name    | Type  |      Owner
--------+-----------+-------+-----------------
 public | employees | table | cmpt354_a3_user
 public | sales     | table | cmpt354_a3_user
 public | types     | table | cmpt354_a3_user
```

---

2. Insert the above records into the tables using INSERT statements

- Employees Relation
  - INSERT INTO employees (eid, name, salary, dept)
    VALUES ('111', 'Jane', 8000, 'Household'),
    ('222', 'Anderson', 8000, 'Toy'),
    ('333', 'Morgan', 10000, 'Cosmetics'),
    ('444', 'Lewis', 12000, 'Stationery'),
    ('555', 'Nelson', 6000, 'Toy'),
    ('666', 'Hoffman', 16000, 'Cosmetics');

```
cmpt354_a3=> SELECT * FROM employees
cmpt354_a3-> ;
        eid          |         name         |         dept         | salary
---------------------+----------------------+----------------------+--------
 111                 | Jane                 | Household            |   8000
 222                 | Anderson             | Toy                  |   8000
 333                 | Morgan               | Cosmetics            |  10000
 444                 | Lewis                | Stationery           |  12000
 555                 | Nelson               | Toy                  |   6000
 666                 | Hoffman              | Cosmetics            |  16000
(6 rows)
```

- Sales relation
  - INSERT INTO Sales (dept, item)
    VALUES ('Stationery', 'pen'),
    ('Cosmetics', 'lipstick'),
    ('Toy', 'puzzle'),
    ('Stationery', 'ink'),
    ('Household', 'disk''),
    ('Sports', 'skates'),
    ('Toy', 'lipstick');

```
cmpt354_a3=> SELECT * FROM sales;
        dept          |         item
----------------------+--------------
 Stationery           | pen
 Cosmetics            | lipstick
 Toy                  | puzzle
 Stationery           | ink
 Household            | disk
 Sports               | skates
 Toy                  | lipstick
(7 rows)
```

- Types relation
  - INSERT INTO Types (item, color)
    VALUES ('pen', 'red'),
    ('lipstick', 'red'),
    ('pen', 'black'),
    ('puzzle', 'black'),
    ('ink', 'red'),
    ('ink', 'blue');

```
cmpt354_a3=> SELECT * FROM types;
        item          |        color
----------------------+--------------
 pen                  | red
 lipstick             | red
 pen                  | black
 puzzle               | black
 ink                  | red
 ink                  | blue
(6 rows)
```

---

3. (42 marks, 6 marks each): Compute the answers to the following queries using SELECT statements. Your SQL statements should be correct for ALL instances of data, not just for the above instance. For example, to find the departments that have a larger average salary than that of "Stationery" department, we do not accept the SQL that uses 12000 as the average salary of "Stationery" department because it only works for the above instance.

1) Compute the maximum salary for each department that sells at least two distinct items.
   - SELECT MAX(E.salary)
     FROM Employees E, Sales S
     WHERE E.dept = S.dept
     GROUP BY E.dept
     HAVING COUNT(DISTINCT S.item) >= 2;

```
cmpt354_a3=> SELECT MAX(E.salary)
cmpt354_a3-> FROM Employees E, Sales S
cmpt354_a3-> WHERE E.dept = S.dept
cmpt354_a3-> GROUP BY E.dept
cmpt354_a3-> HAVING COUNT(DISTINCT S.item) >= 2;
  max
-------
 12000
  8000
(2 rows)
```

2) Compute the names of the employees who work in a department that sells some item in black color
   - SELECT E.name
     FROM Employees E, Sales S, Types T
     WHERE E.dept=S.dept AND S.item=T.item AND T.color='black';

```
cmpt354_a3=> SELECT E.name
cmpt354_a3-> FROM Employees E, Sales S, Types T
cmpt354_a3-> WHERE E.dept=S.dept AND S.item=T.item AND T.color='black';
        name
---------------------
 Anderson
 Lewis
 Nelson
(3 rows)
```

3) For each department that has a larger average salary than that of" Stationery" department, find its average salary.

- SELECT AVG(E.salary)
  FROM Employees E
  WHERE E.dept != 'Stationery"
  GROUP BY E.dept
  HAVING AVG(E.salary) > (SELECT AVG(E2.Salary)
  FROM Employees E2
  WHERE E2.dept = 'Stationery');

```
cmpt354_a3=> SELECT AVG(E.salary)
cmpt354_a3-> FROM Employees E
cmpt354_a3-> WHERE E.dept != 'Stationery'
cmpt354_a3-> GROUP BY E.dept
cmpt354_a3-> HAVING AVG(E.salary) > ( SELECT AVG(E2.Salary)
cmpt354_a3(> FROM Employees E2
cmpt354_a3(> WHERE E2.dept = 'Stationery' );
          avg
------------------------
 13000.0000000000000000
(1 row)
```

4) Find the number of the departments that have a smaller average salary than that of" Stationery" department.

- SELECT COUNT (DISTINCT E.dept)
  FROM Employees E
  WHERE E.dept != 'Stationery'
  GROUP BY E.dept
  HAVING AVG(E.salary) < (SELECT AVG(E2.Salary)
  FROM Employees E2
  WHERE E2.dept = 'Stationery');

```
cmpt354_a3=> SELECT COUNT(DISTINCT E.dept)
cmpt354_a3-> FROM Employees E
cmpt354_a3-> WHERE E.dept != 'Stationery'
cmpt354_a3-> GROUP BY E.dept
cmpt354_a3-> HAVING AVG(E.salary) < ( SELECT AVG(E2.Salary)
cmpt354_a3(> FROM Employees E2
cmpt354_a3(> WHERE E2.dept = 'Stationery' );
 count
-------
     1
     1
```

5) Which department pays every of its employees at least 7000?

- SELECT DISTINCT E.dept
  FROM Employees E
  WHERE NOT EXISTS (
  SELECT *
  FROM Employees E2
  WHERE E2.dept = E.dept AND E2.salary < 7000
  );

```
cmpt354_a3=> SELECT DISTINCT E.dept
cmpt354_a3-> FROM Employees E
cmpt354_a3-> WHERE NOT EXISTS (
cmpt354_a3(>     SELECT *
cmpt354_a3(>     FROM Employees E2
cmpt354_a3(>     WHERE E2.dept = E.dept AND E2.salary < 7000
cmpt354_a3(> );
        dept
----------------------
 Cosmetics
 Stationery
 Household
(3 rows)
```

6) Which departments sell all items sold by Cosmetics department.
   - SELECT DISTINCT S.dept
     FROM Sales S
     WHERE S.dept != ‘Cosmetics’ AND NOT EXISTS (
       SELECT T.item
       FROM Types T, Sales S2
       WHERE S2.dept = 'Cosmetics' AND T.item=S2.item
       EXCEPT
       SELECT T2.item
       FROM Types T2, Sales S3
       WHERE S3.dept = S.dept AND T2.item=S3.item
     );

```
cmpt354_a3=> SELECT DISTINCT S.dept
cmpt354_a3-> FROM Sales S
cmpt354_a3-> WHERE S.dept != 'Cosmetics' AND NOT EXISTS (
cmpt354_a3(>     SELECT T.item
cmpt354_a3(>     FROM Types T, Sales S2
cmpt354_a3(>     WHERE S2.dept = 'Cosmetics' AND T.item=S2.item
cmpt354_a3(>     EXCEPT
cmpt354_a3(>     SELECT T2.item
cmpt354_a3(>     FROM Types T2, Sales S3
cmpt354_a3(>     WHERE S3.dept = S.dept AND T2.item=S3.item
cmpt354_a3(> );
        dept
----------------------
 Toy
(1 row)
```

Part 2: FD & Normalization

1. **(10 marks, 5 marks for correct instances and 5 marks for discussion).** Consider a relation R=(S, A, C, D, T), representing that the student (S) has the address (A), takes the course (C) from the teacher (T) who is from the dept (D). Assume the following FDs, F hold on R:
    - C -> T: a course determines its teacher, i.e., each course is taught by only one teacher
    - S -> A: a student determines its address, i.e., each student has only one address
    - T -> D: a teacher determines its department, i.e., each teacher is from only one department.

However, a student can take multiple courses, a teacher can teach multiple courses, and each department can have multiple teachers. Use an instance of R to explain data redundancy, update anomaly, insertion anomaly, and deletion anomaly that may exist for the above schema R and FDs.

S	A	C	D	T
1	Address1	101	XYZ	Teacher1
1	Address1	102	XYZ	Teacher1
2	Address2	101	XYZ	Teacher1
3	Address3	103	ABC	Teacher2

Key = {Student, Course}

1) **Data Redundancy**
    - Redundancy occurs when the same data is stored in multiple places. Here, we can see that the address and department information is repeated for the same student (S) or teacher (T) in multiple rows.
    - For example, the address "Address1" and department "XYZ" are repeated for the student with ID 1, leading to redundancy.
2) **Update Anomaly**
    - Update anomalies occur when a change in data must be made in multiple places. In this case, if a student changes their address, you will need to update multiple rows.
    - For example, if the address of student 1 changes from "Address1" to "NewAddress," you would need to update multiple rows with the same student ID.
3) **Insertion Anomaly**
    - Insertion anomalies occur when certain information cannot be added to the database without the presence of other information. In this case, you cannot insert information about a new course without assigning it to a teacher and a department.
    - For example, you cannot insert a new course with a new course ID without knowing the teacher and department, which might not be available until the course has students enrolled.
4) **Deletion Anomaly**
    - Deletion anomalies occur when removing certain data leads to unintended loss of other data. In this case, if you delete information about a student, you may unintentionally lose information about the address
    - Deleting the row with student ID 1 would result in the loss of information about the address "Address1," and, consequently, the teacher and department information for the courses associated with that student

2. (38 marks) Continue with the R and FDs F in Question 1.

1) (3 marks) Find all keys of R with respect to F.

- (Student, Course)

---

2) (3 marks) Test if R in BCNF with respect to F, why?

- C+ = {C, T, D}
- S+ = {S, A}
- T+ = {T, D}
- Since C, S, and T do not individually determine all relations, thus not in BCNF

---

3) (10 marks) Produce a BCNF decomposition through a series of binary decomposition. For each binary decomposition, tell the FD used for the decomposition and show the FDs holding on the decomposed tables.

Violation: C -> T

- C+ = {C, T, D}

Decompose into:

- R1(C, T, D)
  - R1 is in BCNF since C -> T and T -> D
- R2(C, S, A)
  - Not in BSNF since S -> A but nothing determines C: DECOMPOSE
  - R3(C, S)
  - R4(S, A)

Therefore, we have:

- R1(C, T, D)
- R3(C, S)
- R4(S, A)

Where:

- R1 tells us about courses taught by teachers & their dept
- R3 tells us what courses students take
- R4 tells us students addresses

---

4) (3 marks) Explain why the decomposed tables produced in 3 are a better representation than the original single table R.

- Less redundancy by decomposing tables into multiple tables
- Avoid update, deletion, and insertion anomalies

---

5) (3 marks) Is the final decomposition in 3 dependency-preserving, why

- Yes, it is dependency conserving since we still satisfy all the original functional dependencies.

---

6) (3 marks) Is the original schema R in 3NF with respect to F, why

- Since {C, S} are the prime values and they only exist on the left side, further none of the FD's left sides are superkeys individually
- Thus, the relation is not in 3NF

7) (10 marks) If the answer to 6 is no, produce a 3NF decomposition that is lossless and dependency-preserving.

- Minimal Cover = {C -> T, S -> A, T -> D}
- Key = {C, S}
- 3NF Synthesis/Decomposition
    - R1 = {CT}
    - R2 = {SA}
    - R3 = {TD}
    - R4 = {CS} //this is the added key

---

8) (3 marks) Is the decomposition produced in 7 in BCNF?

- R1(CT) has FD of C -> T
    - Is in BCNF since C determines T
- R2(SA) has FD S -> A
    - Is in BCNF since S determines A
- R3(TD) as FD T -> D
    - Is in BCND since T determines D
- R4(CS) both C & S in original key
    - Is in BCNF
- Therefore, the decomposition produced in 7 is in BCNF.