1) a)

```
CREATE TABLE Department (
    dept_id INT,
    location VARCHAR(255),
    name VARCHAR(255)
    emp_id INT NOT NULL DEFAULT 101,
    PRIMARY KEY (dept_id),
    FOREIGN KEY (emp_id) REFERENCES Employee (emp_id)
    ON DELETE SET DEFAULT);
```

```
CREATE TABLE Employee (
   emp_id INT,
   name VARCHAR(255),
   surname VARCHAR(255),
   salary FLOAT,
   gender VARCHAR(255),
   PRIMARY KEY (emp_id));
```

```
CREATE TABLE Reports_to(
    supervisor_emp_id INT,
    subordinate_emp_id INT,
    PRIMARY KEY (supervisor_emp_id, subordinate_emp_id),
    FOREIGN KEY supervisor_emp_id REFERENCES Employee(emp_id),
    FOREIGN KEY subordinate_emp_id REFERENCES Employee(emp_id));
```

```
CREATE TABLE works_in (
  dept_id INT,
  emp_id INT,
  PRIMARY KEY (dept_id, emp_id),
  FOREIGN KEY dept_id REFERENCES Department (dept_id)
  ON DELETE NO ACTION,
  FOREIGN KEY emp_id REFERENCES Employee (emp_id)
  ON DELETE CASCADE);
```

```
CREATE TABLE Project (
  project_id INT,
  state VARCHAR(255),
  due_date DATE,
  budget FLOAT,
  dept_id INT NOT NULL,
  PRIMARY KEY (dept_id, project_id),
  FOREIGN KEY dept_id REFERENCES Department (dept_id),
  ON DELETE CASCADE);
```

b)

```
CREATE ASSERTION Total
  CHECK(
     NOT EXISTS(
          SELECT w.emp_id
          FROM works_in w
          GROUP BY w.emp_id
          HAVING COUNT (w.dept_id)=0
     )
  );
```

c)

```
• CREATE TABLE Employee(
      emp_id INT,
      name VARCHAR(255),
      surname VARCHAR(255),
      salary FLOAT CHECK (salary >= 36000),
      gender VARCHAR(255),
      PRIMARY KEY(emp_id));

• CREATE TABLE Department(
      dept_id INT,
      location VARCHAR(255),
      name VARCHAR(255) CHECK( name LIKE CONCAT ('%', location) OR
                                    name LIKE CONCAT (location, '%'),
      emp_id NOT NULL DEFAULT 101,
      PRIMARY KEY (dept_id),
      FOREIGN KEY (emp_id) REFERENCES Employee(emp_id),
      ON DELETE SET DEFAULT);
```

d)

```
CREATE TRIGGER Trip_project
 AFTER UPDATE
 ON Project
 REFERENCING NEW Row AS New_row
 REFERENCING OLD Row AS old_row
 FOR EACH Row
 WHEN (old_row.budget > New_row.budget)
 UPDATE Project
 SET state = 'Unsuccessful'
 WHERE project_id = Old_row.project_id AND
         dept_id = Old_row.dept_id);
```

2)

- R has maximum (100 rows for products) x (5 rows for stores). Total 500 rows
- R has maximum (990 rows for customers) x (100 rows products for customers) + (100 rows products for sales). Total 99100 rows.

3) a)

(1) CB→F

(2) B→E

(3) FE→G

(4) CB→B Trivial

(5) CB→E Transitivity (2,4)

(6) CB→EF Combination (1,5)

(7) CB→G Transitivity (3,6)

b)

(1) A→C

(2) B→E

(3) CB→F

(4) AB→CE Combination (1,2)

(5) AB→CB Trivial

(6) CB→EF Combination (2,3)

(7) AB→EF Transitivity (5,6)

4) a)

$\{A\}^+ \rightarrow \{A,B\}$

$\{D\}^+ \rightarrow \{C,D,E,G\}$

$\{F\}^+ \rightarrow \{C,D,E,F,G\}$

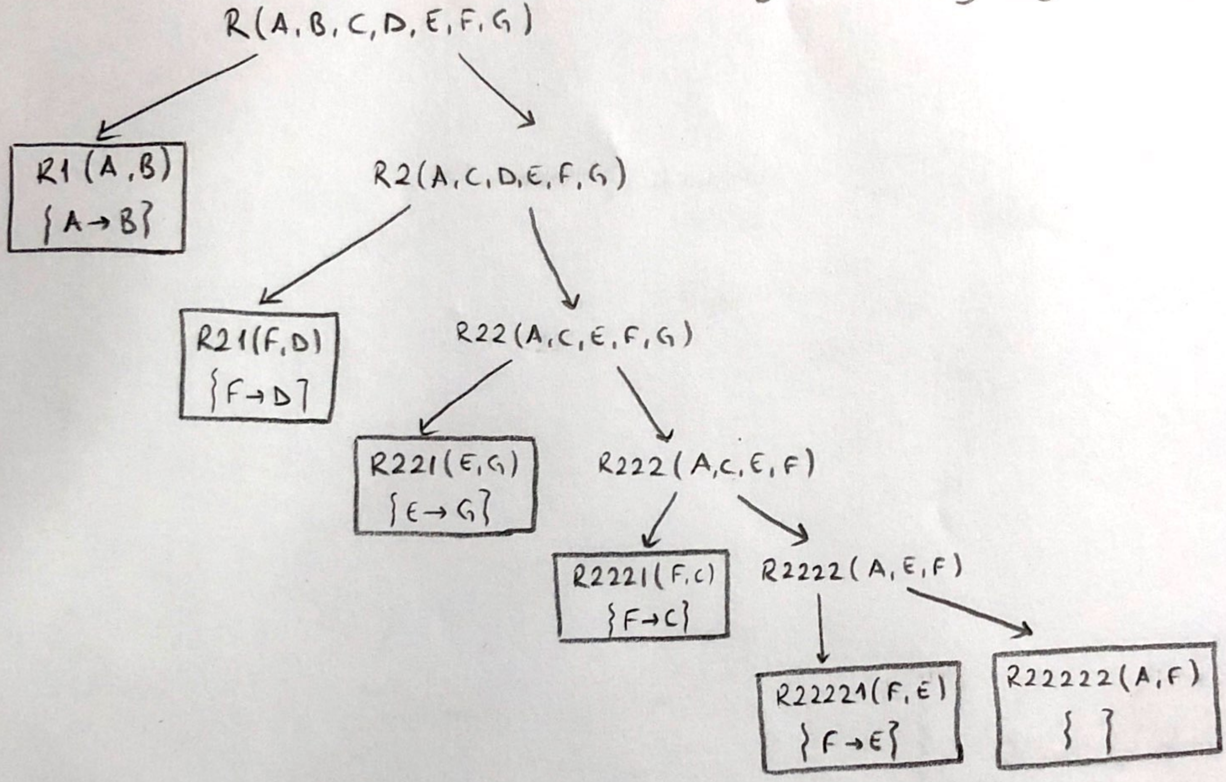
$\{A,C\}^+ \rightarrow \{A,B,C,D,E,G\}$

$\{A,F\}^+ \rightarrow \{A,B,C,D,E,F,G\}$

keys $\Rightarrow \{AF\}^+$ ⟶ The keys must include both A and F, because A and F cannot be trivial by another one. So, all combinations which includes A and F are (super)keys. The only key is $\{AF\}^+$.

b) The left-hand sides of the FDs are not key, and also FDs are non-trivial. So, R is not in BCNF.

c)



d) i)

Following FDs are lost:

$CD \rightarrow E$

$AC \rightarrow D$

$D \rightarrow C$

So, it is not dependency-preserving

ii)

BCNF decomposition is always lossless. So, it is lossless-join.

5) a)

A→E

C→A
C→B
C→E
} we can combine them as C→ABE

E→A

AB→C

BE→C

b)

```
CREATE TABLE R1(
    E VARCHAR(255),
    A VARCHAR(255),
    PRIMARY KEY(E));
```

```
CREATE TABLE R2(
    C INT,
    E VARCHAR(255),
    PRIMARY KEY(C));
```

```
CREATE TABLE R3(
    C INT,
    B VARCHAR(255),
    PRIMARY KEY(C));
```

```
CREATE TABLE R4(
    C INT,
    D INT);
```

c)

```
INSERT INTO R1
    SELECT DISTINCT E,A
    FROM wa1;
```

```
INSERT INTO R2
    SELECT DISTINCT C,E
    FROM wa1;
```

```
INSERT INTO R3
    SELECT DISTINCT C,B
    FROM wa1;
```

```
INSERT INTO R4
    SELECT DISTINCT C,D
    FROM wa1;
```

EXTENSION

In the 5a, when I tried to find FDs I used the following query

```
SELECT [column1]
FROM wa1
GROUP BY [column1]
HAVING COUNT(DISTINCT [column2])>1;
```

By using this query, I saw that if query returns empty table, this means the FD is valid. For example;

```
SELECT A
FROM wa1
GROUP BY A
HAVING COUNT(DISTINCT E) >1;
```

```
SELECT A,B
FROM wa1
GROUP BY A,B
HAVING COUNT (DISTINCT C) >1;
```

These queries returns empty tables, so, the A→E and AB→C FDs are valid.