Task 1 (3 marks)
Clustering

Consider the relational tables created by the execution of the following `CREATE TABLE` statements.

```
CREATE TABLE EMPLOYEE(
ENUM      DECIMAL(12)   NOT NULL,
FNAME     VARCHAR(50)   NOT NULL,
INITIALS  VARCHAR(5)        NULL,
LNAME     VARCHAR(50)   NOT NULL,
DOB       DATE          NULL,
BLDG      DECIMAL(3)    NOT NULL,
STREET    VARCHAR(50)   NOT NULL,
SUBURB    VARCHAR(50)   NOT NULL,
STATE     VARCHAR(5)    NOT NULL,
ZIPCODE   DECIMAL(4)    NOT NULL,
 CONSTRAINT EMPLOYEE_PKEY PRIMARY KEY(ENUM) );

CREATE TABLE DRIVER(
ENUM      DECIMAL(12)   NOT NULL,
LNUM      DECIMAL(8)    NOT NULL,
STATUS    VARCHAR(10)   NOT NULL,
 CONSTRAINT DRIVER_PKEY PRIMARY KEY(ENUM),
 CONSTRAINT DRIVER_UNIQUE UNIQUE(LNUM),
 CONSTRAINT DRIVER_FKEY FOREIGN KEY(ENUM) REFERENCES EMPLOYEE(ENUM),
 CONSTRAINT  DRIVER_STATUS  CHECK  (  STATUS  IN  ('AVAILABLE',  'BUSY',  'ON
LEAVE')) );

CREATE TABLE ADMIN(
ENUM      DECIMAL(12)    NOT NULL,
POSITION  VARCHAR(50)    NOT NULL,
 CONSTRAINT ADMIN_PKEY PRIMARY KEY(ENUM),
 CONSTRAINT ADMIN_FKEY FOREIGN KEY(ENUM) REFERENCES EMPLOYEE(ENUM) );

CREATE TABLE TRUCK(
REGNUM    VARCHAR(10)   NOT NULL,
CAPACITY  DECIMAL(7)    NOT NULL,
WEIGHT    DECIMAL(5)    NOT NULL,
STATUS    VARCHAR(10)   NOT NULL,
 CONSTRAINT TRUCK_PKEY PRIMARY KEY(REGNUM),
 CONSTRAINT TRUCK_STATUS CHECK (STATUS IN ('AVAILABLE', 'USED',
                                                    'MAINTAINED')) );
CREATE TABLE TRIP(
TNUM      DECIMAL(10)   NOT NULL,
LNUM      DECIMAL(8)    NOT NULL,
REGNUM    VARCHAR(10)   NOT NULL,
TRIP_DATE DATE          NOT NULL,
 CONSTRAINT TRIP_PKEY PRIMARY KEY (TNUM),
 CONSTRAINT TRIP_FKEY1 FOREIGN KEY (LNUM) REFERENCES DRIVER(LNUM),
 CONSTRAINT TRIP_FKEY2 FOREIGN KEY (REGNUM) REFERENCES TRUCK(REGNUM) );

CREATE TABLE TRIPLEG(
TNUM      DECIMAL(10)   NOT NULL,
LEGNUM    DECIMAL(2)    NOT NULL,
```

```
DEPARTURE VARCHAR(30)   NOT NULL,
DESTINATION VARCHAR(30) NOT NULL,
CONSTRAINT TRIPLEG_PKEY PRIMARY KEY (TNUM, LEGNUM),
 CONSTRAINT TRIPLEG_UNIQUE UNIQUE(TNUM, DEPARTURE, DESTINATION),
 CONSTRAINT TRIPLEG_FKEY1 FOREIGN KEY (TNUM) REFERENCES TRIP(TNUM) );
```

The database contains information about employees, drivers and administration staff, trucks, trips made by drivers, and legs of each trip.

After loading data into the database the relational tables have the following sizes:

EMPLOYEE	60 data blocks
DRIVER	30 data blocks
ADMIN	10 data blocks
TRUCK	50 data blocks
TRIP	100 data blocks
TRIPLEG	300 data blocks

We would like to use clustering to improve performance of the following types of queries:

(i) Find full information about the drivers who live at a given address.
(ii) Find full information about the administration people who live at a given address.
(iii) Find full information about the trucks used by a driver with a given license number.
(iv) Find full information about the drivers who made a trip on a given date.
(v) Find full information about the legs of trips that used a truck with a given registration number.

Assume, that queries *(i)* and *(ii)* are processed 10 times per day. Assume that queries *(iii)* and *(iv)* are processed 30 times per day. Assume that query *(v)* is processed 20 times per day.

Assume that the relational tables `r` and `s` consist of b_r and b_s blocks each. Then

- if `r` and `s` are clustered together then to read a cluster we need $b_r + b_s$ read block operations and
- if `r` and `s` are not clustered together then to join the tables we need $3*(b_r + b_s)$ read block operations (approximate estimation of hash-based join).

Use a method of finding suboptimal clustering explained to you during the lecture classes in a presentation 18 Clustering to find suboptimal clustering of the sample database that improves the performance of the queries listed above.

For each one of the queries listed above find all joins of the relational tables that must be done to process a query.

(i) Find full information about the drivers who live at a given address.

(ii) Find full information about the administration people who live at a given address.
company.

(iii) Find full information about the trucks used by a driver with a given license number.

(iv) Find full information about the drivers who made a trip on a given date.

(v) Find full information about the legs of trips that used a truck with a given registration number.

Assume, that the queries *(1)* and *(2)* are processed 20 times per day. Assume that the queries *(3)* and *(4)* are processed 10 times per day. Assume that a query *(5)* is processed 5 times per day.

Assume *hash based* implementation of join operation. It means that if the relational tables `r` and `s` consist of b_r and b_s blocks then their sequential scan requires b_r and b_s read block operations and their join, i.e. `r JOIN s` requires $3 * (b_r + b_s)$ read block operations.

When more than 2 tables are joined in a query, consider such query as a sequence of binary join operations. Then, an order of join operations is up to you.

Use a method of finding suboptimal clustering explained to you during the lecture classes in a presentation *18 Clustering* to find suboptimal clustering of the sample database that improves the performance of the queries listed above.

(i) Find full information about the drivers who live at a given address.

```
DRIVER JOIN EMPLOYEE
```

(ii) Find full information about the administration people who live at a given address.

```
ADMIN JOIN EMPLOYEE
```

(iii) Find full information about the trucks used by a driver with a given license number.

```
TRUCK JOIN TRIP
```

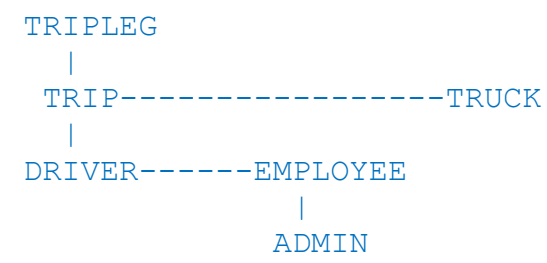
(iv) Find full information about the drivers who made a trip on a given date.

```
DRIVER JOIN TRIP
```

(v) Find full information about the legs of trips that used atruck with a given registration number.

```
TRIPLEG JOIN TRIP
```

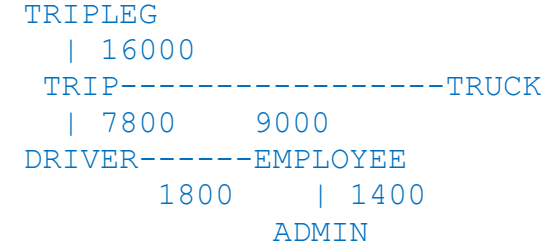
Clustering graph without labels



Profits

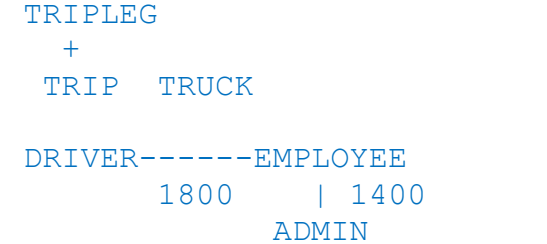
```
DRIVER JOIN EMPLOYEE ===>
10 * 3 * ( 30 + 60 ) - ( 30 + 60 )  = 20*90 = 1800

ADMIN JOIN EMPLOYEE  ===>
10 * 3 * ( 10 + 60 ) - ( 10 + 60 )  = 20*70 = 1400

TRUCK JOIN TRIP      ===>
30 * 3 * ( 50 + 100) - ( 50 + 100 ) = 60*150 = 9000

DRIVER JOIN TRIP     ===>
30 * 3 * ( 30 + 100) - ( 30 + 100 ) = 60*130 = 7800

TRIPLEG JOIN TRIP    ===>
 20 * 3 * ( 300 + 100)- ( 300+ 100 ) = 40*400 = 16000
```

Clustering graph with labels



Step 1

Cluster: `TRIP + TRIPLEG`



Step 2
Cluster: DRIVER + EMPLOYEE

```
TRIPLEG
  +
 TRIP  TRUCK

DRIVER + EMPLOYEE

             ADMIN
```

End of sample solution