Assignment 2

Group 11:

Robert Locher (5747465)**, Jose Enrique Leal Castillo** (9066381)**, Niek Lieon** (6520448)

TASK1)

#First: classroom and department

```
table classroom
CREATE TABLE classroom(
  building VARCHAR(100),
  room_no INTEGER,
  capacity INTEGER,
  PRIMARY KEY (building, room_no));

table department
CREATE TABLE department(
  dept_name VARCHAR(100),
  building VARCHAR(100),
  budget DOUBLE,
  PRIMARY KEY (dept_name));
```

#Second: student, instructor and course

```
table student
CREATE TABLE student(
  ID INTEGER,
  name VARCHAR(100),
  dept_name VARCHAR(100),
  tot_cred double,
  PRIMARY KEY (ID),
  FOREIGN KEY (dept_name) REFERENCES department);

table instructor
CREATE TABLE instructor(
  ID INTEGER,
  name VARCHAR(100),
  dept_name VARCHAR(100),
  salary DOUBLE,
  PRIMARY KEY (ID)
  FOREIGN KEY (dept_name) REFERENCES department);
```

```
table course
CREATE TABLE course(
  course_id INT,
  title VARCHAR(100),
  dept_name VARCHAR(100),
  credits DOUBLE,
  PRIMARY KEY (course_id),
  FOREIGN KEY (dept_name) REFERENCES department);
```

#Third: advisor, section, and prereq

```
table advisor
CREATE TABLE advisor(
  s_id INTEGER,
  i_id INTEGER,
  PRIMARY KEY (s_id),
  FOREIGN KEY (s_id) REFERENCES student,
  FOREIGN KEY (i_id) REFERENCES instructor);

table section
CREATE TABLE section(
  course_id INTEGER,
  sec_id INTEGER,
  semester VARCHAR(100) CHECK (semester IN ('Q1', 'Q2' ,'Q3', 'Q4')),
  year INTEGER CHECK (year > 2000),
  building VARCHAR(100),
  room_no INTEGER,
  time_slot_id VARCHAR(100) CHECK (time_slot_id IN ('A', 'B', 'C', 'D')),
  PRIMARY KEY (course_id, sec_id, semester, year),
  FOREIGN KEY (Building, room_no) REFERENCES classroom,
  FOREIGN KEY (course_id) REFERENCES section);

table prereq
CREATE TABLE prereq(
  course_id INTEGER,
  prereq_id INTEGER,
  PRIMARY KEY (course_id, prereq_id),
  FOREIGN KEY (prereq_id, course_id) REFERENCES course);
```

#Fourth: takes and teaches

table takes
```
CREATE TABLE takes(
  ID INTEGER,
  course_id INTEGER,
  sec_id INTEGER,
  semester INTEGER CHEck (semester IN ('Q1' , 'Q2' ,'Q3' ,'Q4')),
  year INTEGER CHECK (year > 2000),
  Grade DOUBLE,
  PRIMARY KEY (ID, course_id, sec_id, semester, year),
  FOREIGN KEY (ID) REFERENCES student,
  FOREIGN KEY (course_id, sec_id, semester, year) REFERENCES section);
```

table teaches
```
CREATE TABLE teaches(
  ID INTEGER,
  course_id INTEGER,
  sec_id INTEGER,
  semester INTEGER CHECK (semester IN ('Q1' , 'Q2' ,'Q3' ,'Q4')),
  year INTEGER CHECK (year > 2000),
  PRIMARY KEY (ID, course_id, sec_id, semester, year),
  FOREIGN KEY (ID, course_id, sec_id, semester, year) REFERENCES section,
  FOREIGN KEY (ID) REFERENCES instructor);
```

TASK2)

```
INSERT into student
VALUES(13, "Brian","Luck Science","59.5");

INSERT into student
VALUES(666, "Dev","Satanism","66.6");

INSERT into student
VALUES(2, "Binar","Bool Science","58.5");

INSERT into student
VALUES(5, "Penta","Geometry","30");

INSERT into advisor
VALUES(13, 342);

INSERT into advisor
VALUES(666, 999);

INSERT into advisor
VALUES(2, 7474);
```

```
INSERT into advisor
VALUES(5, 23);

INSERT into instructor
VALUES(999, "Tietsjor", "Satanism", 99999.9);

INSERT into instructor
VALUES(342, "HectorLector", "Luck Science", 50000);

INSERT into instructor
VALUES(7474, "Peter", "Bool Science", 99999.9);

INSERT into instructor
VALUES(23, "Jan Modaal", "Geometry", 30000);

SELECT student.name as "Student name", instructor.name as "Instructor name" FROM student
JOIN advisor on (student.ID = advisor.s_id) JOIN instructor on (instructor.ID = advisor.i_id);
```

TASK3)

The candidate key for R is AB. These are the only attributes that are not noted on the RHS of the list of FD's and all other attributes are noted on the RHS. Furthermore, all other attributes can be found when AB are known. The following functional dependencies are known: F = ABD → EG, C → DG, E → FG, AB → C, G → F. The proof that AB is the candidate key can be seen in the table:

Known Attributes	Functional Dependency
{A,B}	Triviality
{A,B,C}	AB → C
{A,B,C,D,G}	C → DG
{A,B,C,D,G,E}	ABD → EG
{A,B,C,D,G,E,F}	G → F

TASK4)

```
import pandas as pd

#Read csv
df = pd.read_csv("fdExample.csv")

#Getting a full list of all names
names = df["First name"].value_counts()
names_list = list(names.index)
names_list
```

```
#Getting prefered gender for each name in a list
names_gender = df.groupby(["First name", "Gender"]).count()

gender_list = []
for n in names_list:
   tab1 = names_gender.loc[n,"Department"]
   tab = names_gender.loc[n,"Department"].idxmax()
   gender_list.append(tab)

#Creating a dict which each name and gender
dic = dict(zip(names_list,gender_list))
dic
test = df.copy()
#List of males names
males=[k for k, v in dic.items() if v == "M"]
#List of females names
females = [k for k, v in dic.items() if v == "F"]

#test female vio
test["Violation male 1"] = (~test["First name"].isin(males) )
test["Violation male 2"] = (test["Gender"] != "M")
test["Violation male"] = (test["Violation male 1"] & test["Violation male 2"] )

#test male vio
test["Violation female 1"] = (~test["First name"].isin(females) )
test["Violation female 2"] = (test["Gender"] != "F")
test["Violation female"] = (test["Violation female 1"] & test["Violation female 2"] )


#Localize violation in a row
test["Violation"] = (test["Violation female"] == test["Violation male"] )

#Finding names in which there is a violation
test.loc[test["Violation"]== True][["First name", "Gender", "Violation"]]
```

TASK5)

The used dataset for this task was the chinook dataset, which can be found on https://www.sqlitetutorial.net/wp-content/uploads/2018/03/chinook.zip

The used algorithm for finding the functional dependencies was TANE, which is described on https://www.lri.fr/~pierres/donn%E9es/save/these/articles/lpr-queue/huhtala99tane.pdf and retrieved from https://github.com/nabihach/FD_CFD_extraction. Because the algorithm only works for tables with column names consisting of the characters A-Z, the names of the tables were changed to these characters. The changes for each table can be found under the corresponding screenshot.

Assignment 2

table album:

```
(base) C:\Users\rober\Downloads\FD_CFD_extraction-master>python tane.py albums.csv
List of all FDs:  [['A', 'C'], ['A', 'B'], ['BC', 'A']]
Total number of FDs found:  3
```

A = AlbumId, B = Title, C = ArtistId

table artists:

```
(base) C:\Users\rober\Downloads\FD_CFD_extraction-master>python tane.py artists.csv
List of all FDs:  [['A', 'B;;']]
Total number of FDs found:  1
```

A = ArtistId, B = Name

table customers:

```
(base) C:\Users\rober\Downloads\FD_CFD_extraction-master>python tane.py customers.csv
List of all FDs:  [['A', 'J'], ['A', 'C'], ['A', 'H'], ['A', 'G'], ['A', 'E'], ['A', 'I'], ['A', 'K'], ['A', 'B'], [
'A', 'L'], ['A', 'F'], ['A', 'M'], ['A', 'D'], ['C', 'B'], ['E', 'B'], ['L', 'B'], ['C', 'D'], ['E', 'C'], ['C', 'E'
], ['C', 'F'], ['C', 'G'], ['C', 'H'], ['C', 'I'], ['C', 'J'], ['C', 'K'], ['L', 'C'], ['C', 'L'], ['E', 'D'], ['I',
 'D'], ['J', 'D'], ['K', 'D'], ['L', 'D'], ['E', 'F'], ['E', 'G'], ['E', 'H'], ['E', 'I'], ['E', 'J'], ['E', 'K'], [
'L', 'E'], ['E', 'L'], ['F', 'G'], ['F', 'H'], ['L', 'F'], ['J', 'G'], ['L', 'G'], ['L', 'H'], ['I', 'K'], ['L', 'I'
], ['J', 'K'], ['L', 'J'], ['L', 'K'], ['BM', 'J'], ['BM', 'C'], ['BM', 'H'], ['BM', 'G'], ['BM', 'E'], ['BM', 'I'],
 ['BM', 'K'], ['BM', 'L'], ['BM', 'F'], ['BM', 'A'], ['BM', 'D'], ['IM', 'J'], ['IM', 'C'], ['IM', 'H'], ['IM', 'G']
, ['IM', 'E'], ['IM', 'B'], ['IM', 'L'], ['IM', 'F'], ['IM', 'A'], ['JM', 'C'], ['JM', 'H'], ['JM', 'E'], ['JM', 'I'
], ['JM', 'B'], ['JM', 'L'], ['JM', 'F'], ['JM', 'A'], ['LM', 'A'], ['BD', 'C'], ['BF', 'C'], ['BG', 'C'], ['BI', 'C
'], ['BJ', 'C'], ['BK', 'C'], ['BD', 'E'], ['BF', 'D'], ['BD', 'F'], ['BG', 'D'], ['BD', 'G'], ['BD', 'H'], ['BD', '
I'], ['BD', 'J'], ['BD', 'K'], ['BD', 'L'], ['BF', 'E'], ['BG', 'E'], ['BI', 'E'], ['BJ', 'E'], ['BK', 'E'], ['BG',
'F'], ['FI', 'B'], ['BI', 'F'], ['BF', 'I'], ['FJ', 'B'], ['BJ', 'F'], ['BF', 'J'], ['BK', 'F'], ['BF', 'K'], ['BF',
 'L'], ['BG', 'H'], ['BI', 'G'], ['BG', 'I'], ['BG', 'J'], ['BK', 'G'], ['BG', 'K'], ['BG', 'L'], ['HI', 'B'], ['BI'
, 'H'], ['HJ', 'B'], ['BJ', 'H'], ['BK', 'H'], ['IJ', 'B'], ['BJ', 'I'], ['BI', 'J'], ['BK', 'I'], ['BI', 'L'], ['BK
', 'J'], ['BJ', 'L'], ['BK', 'L'], ['FI', 'C'], ['FJ', 'C'], ['HI', 'C'], ['HJ', 'C'], ['IJ', 'C'], ['DF', 'K'], ['D
G', 'K'], ['FI', 'E'], ['FJ', 'E'], ['HI', 'E'], ['HJ', 'E'], ['IJ', 'E'], ['HI', 'F'], ['HJ', 'F'], ['IJ', 'F'], ['
FJ', 'I'], ['FI', 'J'], ['FI', 'L'], ['FJ', 'L'], ['HI', 'G'], ['IJ', 'H'], ['HJ', 'I'], ['HI', 'J'], ['HI', 'L'], [
'HJ', 'L'], ['IJ', 'L']]
Total number of FDs found:  156
```

A = CustomerId, B = FirstName, C = LastName, D = Company, E= Address, F = City, G = State, H = Country, I = PostalCode, J = Phone, K = Fax, L = Email, M = SupportRepId

table employees:

```
(base) C:\Users\rober\Downloads\FD_CFD_extraction-master>python tane.py employees.csv
List of all FDs:  [['A', 'N'], ['A', 'C'], ['A', 'L'], ['A', 'K'], ['A', 'I'], ['A', 'D'], ['A', 'H'], ['A', 'G'], [
'A', 'J'], ['A', 'O'], ['A', 'F'], ['A', 'E'], ['A', 'M'], ['A', 'B'], ['B', 'N'], ['B', 'A'], ['B', 'C'], ['B', 'L'
], ['B', 'K'], ['B', 'I'], ['B', 'D'], ['B', 'H'], ['B', 'G'], ['B', 'J'], ['B', 'O'], ['B', 'F'], ['B', 'E'], ['B',
 'M'], ['C', 'N'], ['C', 'A'], ['C', 'L'], ['C', 'K'], ['C', 'I'], ['C', 'D'], ['C', 'H'], ['C', 'G'], ['C', 'J'], [
'C', 'O'], ['C', 'F'], ['C', 'E'], ['C', 'M'], ['C', 'B'], ['F', 'N'], ['F', 'A'], ['F', 'C'], ['F', 'L'], ['F', 'K'
], ['F', 'I'], ['F', 'D'], ['F', 'H'], ['F', 'G'], ['F', 'J'], ['F', 'O'], ['F', 'E'], ['F', 'M'], ['F', 'B'], ['H',
 'N'], ['H', 'A'], ['H', 'C'], ['H', 'L'], ['H', 'K'], ['H', 'I'], ['H', 'D'], ['H', 'G'], ['H', 'J'], ['H', 'O'], [
'H', 'F'], ['H', 'E'], ['H', 'M'], ['H', 'B'], ['L', 'N'], ['L', 'A'], ['L', 'C'], ['L', 'K'], ['L', 'I'], ['L', 'D'
], ['L', 'H'], ['L', 'G'], ['L', 'J'], ['L', 'O'], ['L', 'F'], ['L', 'E'], ['L', 'M'], ['L', 'B'], ['N', 'A'], ['N',
 'C'], ['N', 'L'], ['N', 'K'], ['N', 'I'], ['N', 'D'], ['N', 'H'], ['N', 'G'], ['N', 'J'], ['N', 'O'], ['N', 'F'], [
'N', 'E'], ['N', 'M'], ['N', 'B'], ['O', 'N'], ['O', 'A'], ['O', 'C'], ['O', 'L'], ['O', 'K'], ['O', 'I'], ['O', 'D'
], ['O', 'H'], ['O', 'G'], ['O', 'J'], ['O', 'F'], ['O', 'E'], ['O', 'M'], ['O', 'B'], ['D', 'E'], ['D', 'I'], ['D',
 'J'], ['D', 'K'], ['E', 'I'], ['E', 'J'], ['E', 'K'], ['G', 'I'], ['G', 'J'], ['G', 'K'], ['I', 'J'], ['I', 'K'], [
'M', 'I'], ['K', 'J'], ['J', 'K'], ['M', 'J'], ['M', 'K'], ['DG', 'N'], ['DG', 'A'], ['DG', 'C'], ['DG', 'L'], ['DG'
, 'H'], ['DG', 'O'], ['DG', 'F'], ['DG', 'M'], ['DG', 'B'], ['DM', 'N'], ['DM', 'A'], ['DM', 'C'], ['DM', 'L'], ['DM
', 'H'], ['DM', 'G'], ['DM', 'O'], ['DM', 'F'], ['DM', 'B'], ['EG', 'N'], ['EG', 'A'], ['EG', 'C'], ['EG', 'L'], ['E
G', 'D'], ['EG', 'H'], ['EG', 'O'], ['EG', 'F'], ['EG', 'M'], ['EG', 'B'], ['EM', 'N'], ['EM', 'A'], ['EM', 'C'], ['
EM', 'L'], ['EM', 'D'], ['EM', 'H'], ['EM', 'G'], ['EM', 'O'], ['EM', 'F'], ['EM', 'B'], ['GM', 'N'], ['GM', 'A'], [
'GM', 'C'], ['GM', 'L'], ['GM', 'D'], ['GM', 'H'], ['GM', 'O'], ['GM', 'F'], ['GM', 'E'], ['GM', 'B']]
Total number of FDs found:  177
```

A = EmployeeId, B = LastName, C = FirstName, D = Title, E = ReportsTo, F = BirthDate, G = HireDate, H = Address, I = City, J = State, K = Country, L = PostalCode, M = Phone, N = Fax, O = Email

table genres:

```
(base) C:\Users\rober\Downloads\FD_CFD_extraction-master>python tane.py genres.csv
List of all FDs:  [['A', 'B'], ['B', 'A']]
Total number of FDs found:  2
```

A = GenreId, B = Name

table invoice_items:

```
(base) C:\Users\rober\Downloads\FD_CFD_extraction-master>python tane.py invoice_items.csv
List of all FDs:  [['A', 'E'], ['A', 'C'], ['A', 'B'], ['A', 'D'], ['B', 'E'], ['C', 'D'], ['C', 'E'], ['D', 'E'], [
'BC', 'A']]
Total number of FDs found:  9
```

A = InvoiceLineId, B = InvoiceId, C = TrackId, D = UnitPrice, E = Quantity

table invoices:

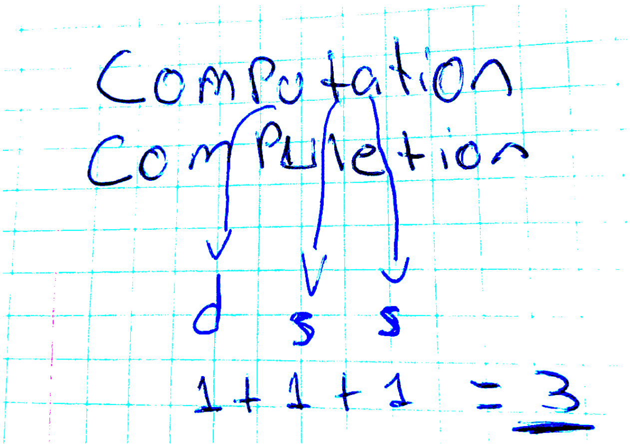
```
(base) C:\Users\rober\Downloads\FD_CFD_extraction-master>python tane.py invoices.csv
List of all FDs:  [['A', 'D'], ['A', 'I'], ['A', 'H'], ['A', 'G'], ['A', 'E'], ['A', 'F'], ['A', 'B'], ['A', 'C'], [
'B', 'D'], ['B', 'E'], ['B', 'F'], ['B', 'G'], ['B', 'H'], ['D', 'E'], ['D', 'F'], ['D', 'G'], ['D', 'H'], ['E', 'F'
], ['E', 'G'], ['BC', 'I'], ['BC', 'A'], ['DI', 'B'], ['CH', 'D'], ['CH', 'E'], ['CH', 'F'], ['CH', 'G'], ['EH', 'D'
], ['GH', 'D'], ['GH', 'E'], ['GH', 'F'], ['CHI', 'B'], ['CHI', 'A'], ['EHI', 'B'], ['FHI', 'B'], ['GHI', 'B'], ['FH
I', 'D'], ['FHI', 'E'], ['FHI', 'G']]
Total number of FDs found:  38
```

A = InvoiceId, B = CustomerId, C = InvoiceDate, D = BillingAddress, E = BillingCity, F = BillingState, G = BillingCountry, H = BillingPostalCode, I = Total

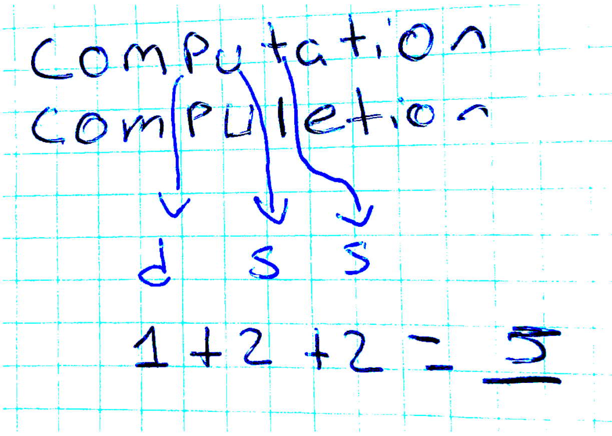
TASK6)

Compute the Levenshtein distance between “Computation” and “Completion” assuming the following costs of the operations:

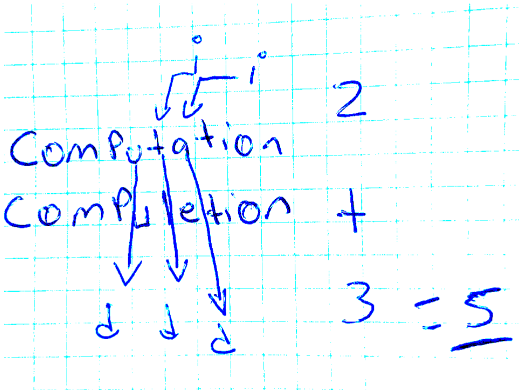
1) Each operation cost 1



2) Update (substitute) costs 2, other operations have cost of 1.

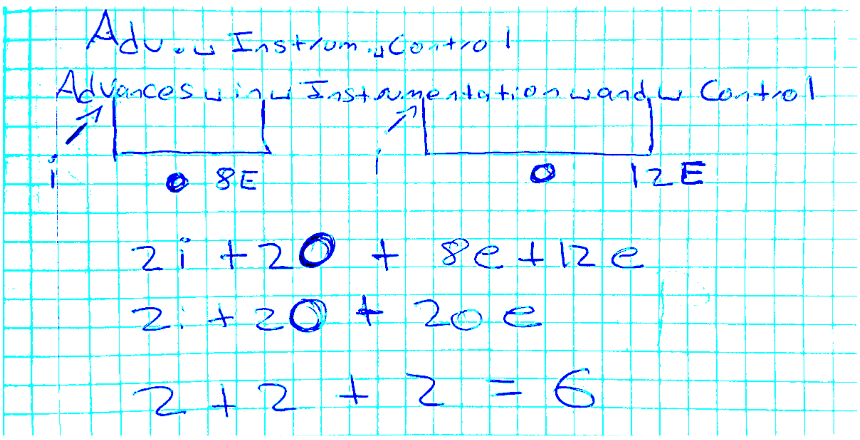


3) Update (substitute) costs 3, other operations have cost of 1.



TASK7)

Compute the gap distance between “Advances in Instrumentation and Control” and “Adv. Instrum. Control” assuming that (insertion cost =open gap cost = 1) and extend gap cost is 0.1.



TASK8)

Compute Jaccard Distance between each pair of the following three sets:A={0,1,2,5,6}; B={0,2,3,5,7,9}; C={2,3,5,6}.

Jaccard distance

- (A,B) = 1 - 3/8 = 5/8
- (A,C) = 1 - 3/6 = 1/2
- (B,C) = 1 - 3/7 = 4/7

TASK9)

A = {1,1,2,2,5};
B = {1,2,2,2,5,5};
C = {1,2,3,4,5}.
U = {1,2,3,4,5}

Jaccard distance

- (A,B) = 1 - 1 = 0
- (A,C) = 1 - 3/5 = 2/5
- (B,C) = 1 - 3/5 = 2/5

Jaccard bag similarity

A∩B = {1,2,2,5}.
- = 4/11

A∩C = {1,2,5}
- = 3/10

B∩C = {1,2,5}
- = 3/11

TASK10)

Compute the Jaro and Jaro-Winkler similarity between arnab and urban. What do you think is the reason for this result?

10

$$\left[\frac{\max(|S_1|, |S_2|)}{2}\right] - 1$$

arnab
urban

$$\frac{5}{2} - 1 = 1.5$$

$$\boxed{C = 2} \qquad \boxed{T = \frac{0}{2} = 0}$$

$$\text{Jaro Sim}(S_1, S_2) = \frac{1}{3}\left(\frac{C}{|S_1|} + \frac{C}{|S_2|} + \frac{C-T}{C}\right)$$

$$\text{JaroSim}(S_1, S_2) = \frac{1}{3}\left(\frac{2}{5} + \frac{2}{5} + \frac{2}{2}\right)$$

$$\text{Jaro Sim}(S_1, S_2) = 0.6$$

Jaro Winkler.-

Jaro Winkler(S1,S2) = JaroSim + [P * L * (1 - JaroSim)]

Jaro Winkler(S1,S2) = 0.6 + [0.1 * 0 * (1 - 0.6)] = 0.6

Conclusion: Jaro and Jaro-Winkler similarities are the same for this case because the words are not starting with the same letter, thus the L = 0.

TASK11)

"Many problems can be expressed as finding similar sets"						
"Many-"	"blems"	"n-be-"	"resse"	"s-fin"	"g-sim"	"r-set"
"any-p"	"lems-"	"-be-e"	"essed"	"-find"	"-simi"	"-sets"
"ny-pr"	"ems-c"	"be-ex"	"ssed-"	"findi"	"simil"	
"y-pro"	"ms-ca"	"e-exp"	"sed-a"	"indin"	"imila"	
"-prob"	"s-can"	"-expr"	"ed-as"	"nding"	"milar"	
"probl"	"-can-"	"expre"	"d-as-"	"ding-"	"ilar-"	
"roble"	"can-b"	"xpres"	"-as-f"	"ing-s"	"lar-s"	
"oblem"	"an-be"	"press"	"as-fi"	"ng-si"	"ar-se"	

Cardinality: 50

TASK 12)

(a):

The matrix representation of the documents:

Shingle	D1	D2	D3
aa	1	1	0
bb	1	0	0
ab	1	0	1
ba	1	1	1
ac	0	1	0
ca	0	1	1

(b) Consider the following permutations:p1 =

{aa, bb, ab, ba, ac, ca},p2 ={ca, ac, ba, ab, bb, aa},p3 ={ac, ca, ab, ba, bb, aa}.

Create the signature matrix of the documents

P1	P2	P3
1	6	6
2	5	5
3	4	3
4	3	4
5	2	1
6	1	2

Shingle	D1	D2	D3
aa	1	1	0
bb	1	0	0
ab	1	0	1
ba	1	1	1
ac	0	1	0
ca	0	1	1

Signature matrix)

	D1	D2	D3
P1	1	1	3
P2	3	1	1
P3	3	1	2

(c)

Jaccard similarity:

Similarity	D1 - D2	D1 - D3	D2 -D3
Matrix representation	0.33	0.4	0.4
Signature	0.33	0	0.33

The Jaccard similarities are roughly equal. They have some error which is due to the very small sample set. The law of large numbers tells that as the sample size grows, the difference between the Jaccard similarity of the signature matrix and the documents themselves will even out.