Contents

1 Basic Test Results

```
Extracting Archive:
Archive:  /tmp/bodek.N5LEo7/db/ex5/mohammadgh/presubmission/submission
  inflating: contradictions.sql
  inflating: create.sql
  inflating: drop.sql
  inflating: ex5.pdf
  inflating: README
Presubmission test

*************************************************************
** Testing that all necessary files were submitted:
README:
    SUBMITTED
ex5.pdf:
    SUBMITTED
create.sql:
    SUBMITTED
contradictions.sql:
    SUBMITTED
drop.sql:
    SUBMITTED

*************************************************************
** Checking for correct README format:
Note: The output is capped at 500000 characters.
Inserting sales.csv
Output:
COPY 7999


Running contradictions.sql
Output:
 transactionno | productno
---------------+-----------
 536373        | 20679
 536375        | 20679
 536386        | 84880
 536386        | 85099B
 536386        | 85099C
 536387        | 21731
 536387        | 22466
 536387        | 22779
 536387        | 22780
 536387        | 79321
 536396        | 20679
 536406        | 20679
 536531        | 20679
 536595        | 20679
 536749        | 20679
 536750        | 20679
 536752        | 20679
 536790        | 20679
 536804        | 20679
 536831        | 21928
 536831        | 21929
 536831        | 22207
 536831        | 84880
 536865        | 20679
 536974        | 20679
```

```
 537042         | 20679
 537081         | 20679
(27 rows)
```

2 README

```
mohammadgh,muaz.abdeen
```

3 contradictions.sql

```
select distinct s1.TransactionNo, s1.ProductNo
from sales s1, sales s2
where (s1.TransactionNo = s2.TransactionNo and s1.Date != s2.Date) or
      (s1.TransactionNo = s2.TransactionNo and s1.CustomerNo != s2.CustomerNo) or
      (s1.ProductNo = s2.ProductNo and s1.ProductName != s2.ProductName) or
      (s1.CustomerNo = s2.CustomerNo and s1.Country != s2.Country)
order by s1.TransactionNo, s1.ProductNo asc;
```

4 create.sql

```sql
create table sales
(
    TransactionNo varchar,
    Date          varchar,
    ProductNo     varchar,
    ProductName   varchar,
    Price         varchar,
    Quantity      varchar,
    CustomerNo    varchar,
    Country       varchar
);
```

5 drop.sql

```
drop table if exists sales;
```

(67506) Databases – Spring 2022 – Exercise (3)

Muaz Abdeen 300575297

Mohammad Ghanayem 208653220

Question (1):

Oscars(ID, Title, Year, Studio, Award, releaseYear, Duration, Genres, imdbRating, imdbVotes, contentRating, Directors, Authors, Actors)

a) This relation is not in first normal form, because we have attributes that don't have atomic values like Genres and Authors.
Two problems we could face: <u>Data duplication</u>, in Actors attribute we might face a situation that we have the same actors in different oscars, <u>Update anomaly</u>, when we update for example a value in the Actors attributes, we must update many rows (if there was a duplication).

b) Oscars (ID, Title, Year, Studio, Award, releaseYear, Duration, imdbRating, imdbVotes, contentRating)

F = {ID → Title, Studio, releaseYear, Duration
Title, Studio → Year, Award
Title, Duration, releaseYear → imdbRating, imdbVotes, contentRating
Title, releaseYear → ID, Studio, Year
Title, Year → Duration}

Now, we want to find the Minimal Cover…
We want to use ComputeMinimalCover(F) algorithm to find the minimal cover for F, first, we do the first process in the algorithm (first step), and we got:

G = {ID → Title
ID → Studio
ID → releaseYear
ID → Duration
Title, Studio → Year
Title, Studio → Award
Title, Duration, releaseYear → imdbRating

Title, Duration, releaseYear → imdbVotes
Title, Duration, releaseYear → contentRating
Title, releaseYear → ID
Title, releaseYear → Studio
Title, releaseYear → Year
Title, Year → Duration }

Then we will do the second step…
we got that $Duration \in (Title, releaseYear)^+$because $(Title, releaseYear)$is a key, so we remove "Duration" attribute from each "Title, Duration, releaseYear" dependency, the other dependencies stays as they are, so we got this modified G:

G = {ID → Title
ID → Studio
ID → releaseYear
ID → Duration
Title, Studio → Year
Title, Studio → Award
Title, releaseYear → imdbRating
Title, releaseYear → imdbVotes
Title, releaseYear → contentRating
Title, releaseYear → ID
Title, releaseYear → Studio
Title, releaseYear → Year
Title, Year → Duration}

Now, we do the third and final step of the algorithm…
We want to check if there is any dependency that follows from other dependencies, so we got that (ID → Studio) follows from (ID → Title & ID → releaseYear & (Title, releaseYear)→ Studio) dependencies, and (ID → Duration) follows from (ID → Title & ID → releaseYear & (Title, releaseYear) → ID & (Title, releaseYear) → Year & (Title, Year) → Duration) dependencies, and (Title, releaseYear → Year) follows from (ID → Title & ID → releaseYear & (Title, releaseYear) → ID & (Title, releaseYear) → Studio)) dependencies. So, we got these final dependencies:

G = {ID → Title

ID → releaseYear

Title, Studio → Year

Title, Studio → Award

Title, releaseYear → imdbRating

Title, releaseYear → imdbVotes

Title, releaseYear → contentRating

Title, releaseYear → ID

Title, releaseYear → Studio

Title, Year → Duration}

And this is our Minimal Cover of F, and we done.

c) We can see that our keys are (ID) and (Title, releaseYear), because of the fact that the none of the BCNF and 3NF attributes holds on our relation, we got that our relation is neither BCNF nor 3NF.

d) R1 = (ID, Title, Duration, Award)
R2 = (Title, releaseYear, imdbRating, imdbVotes, contentRating)
R3 = (ID, Studio, Year)
We want to check that if this decomposition is lossless or not, so we use CreateTable(Oscars, (R1, R2, R3)) algorithm to build our table, then we use ChaseTable(T, F) algorithm to find out if our decomposition is lossless or not, the table after running CreateTable algorithm will be:

	ID	Title	Year	Studio	Award	releaseYear	Duration	imdbRating	imdbVotes	contentRating
ID, Title, Duration, Award	a1	a2	b1,3	b1,4	a5	b1,6	a7	b1,8	b1,9	b1,10
Title, releaseYear, imdbRating, ImdbVotes, contentRating	b2,1	a2	b2,3	b2,4	b2,5	a6	b2,7	a8	a9	a10
ID, Studio, Year	a1	b3,2	a3	a4	b3,5	b3,6	b3,7	b3,8	b3,9	b3,10

Now we need to run ChaseTable algorithm to check if our decomposition is lossless or not, the resulted table will be:

	ID	Title	Year	Studio	Award	releaseYear	Duration	imdbRating	imdbVotes	contentRating
ID, Title, Duration, Award	a1	a2	a3	a4	a5	b3,6	a7	b3,8	b3,9	b3,10
Title, releaseYear, imdbRating, ImdbVotes, contentRating	b2,1	a2	b2,3	b2,4	b2,5	a6	b2,7	a8	a9	a10
ID, Studio, Year	a1	a2	a3	a4	a5	b3,6	a7	b3,8	b3,9	b3,10

So, we got that our decomposition is not lossless (with loss).

e) We want to find decomposition for Oscars to 3NF by using the algorithm that we learned in class, we found the minimal cover in section b, now we create the schemas:

R1 = (ID, Title)
R2 = (ID, releaseYear)
R3 = (Title, Studio, Year)
R4 = (Title, Studio, Award)
R5 = (Title, releaseYear, imdbRating)
R6 = (Title, releaseYear, imdbVotes)
R7 = (Title, releaseYear, contentRating)
R8 = (Title, releaseYear, ID)
R9 = (Title, releaseYear, Studio)
R10 = (Title, Year, Duration)

we can see that R1 and R2 are both in R8, so we remove them and we stay with these schemas:

R3 = (Title, Studio, Year)
R4 = (Title, Studio, Award)
R5 = (Title, releaseYear, imdbRating)
R6 = (Title, releaseYear, imdbVotes)
R7 = (Title, releaseYear, contentRating)
R8 = (Title, releaseYear, ID)
R9 = (Title, releaseYear, Studio)
R10 = (Title, Year, Duration)

All of the subschemas are BCNF.

f) Using the given F in the question we run the algorithm FindBCNFDecomposition(R, F):
In the first call of the function R not BCNF, we found out that the dependency $(Title, Studio \rightarrow Year, Award)$ is a BCNF violation, so we divide our R to R1 = (Title, Studio, Year, Award, Duration) and R2 = (Title, Studio, ID, releaseYear, imdbRating, imdbVotes, contentRating), now we recursively call for the function with R1 and F_{R_1}, by trying to define

this R1's dependencies we found out that the dependency $(Title, Year \rightarrow Title, Year, Duration)$ is a BCNF violation so its division will be R3 = (Title, Year, Duration) and R4 = (Title, Year, Studio, Award), we can see that R3 is BCNF and it is our first one, the same for R4, now we return to R2, the same for it too, we finished the run of the algorithm and we got three subschemas and they are:

R2 = (Title, Studio, ID, releaseYear, imdbRating, imdbVotes, contentRating)
R3 = (Title, Year, Duration)
R4 = (Title, Year, Studio, Award)

Now, to see that this decomposition is Dependency Preserving we use the given algorithm in the class (IsDependencyPreserving):

F = {ID → Title, Studio, releaseYear, Duration

Title, Studio → Year, Award

Title, Duration, releaseYear → imdbRating, imdbVotes, contentRating

Title, releaseYear → ID, Studio, Year

Title, Year → Duration}

We can see that the dependencies 2 and 5 preserved because left and right side belongs to the subschemas R4 and R3, respectively.
I will give symbols to each attribute in Oscars, T=Title, Y=Year, S=Studio, A=Award, rY=releaseYear, D=Duration, iR=imdbRating, iV=imdbVotes, cR=contentRating. R1=R2, R2=R3, R3=R4
Now we want to check if the dependencies 1, 3 and 4 are preserved or not, we begin with 1:

$$Z' = ID$$
$$Z = ID \cup ((ID \cap R_1)^+ \cap R_1) = (T, S, ID, rY, iR, iV, cR)$$
$$Z = (T, S, ID, rY, iR, iV, cR) \cup \left(\left((T, S, ID, rY, iR, iV, cR) \cap R_2\right)^+ \cap R_2\right) = (T, S, ID, rY, iR, iV, cR)$$
$$Z = (T, S, ID, rY, iR, iV, cR) \cup \left(\left((T, S, ID, rY, iR, iV, cR) \cap R_3\right)^+ \cap R_3\right) = (T, S, ID, rY, iR, iV, cR, Y, A)$$
$$Z = (T, S, ID, rY, iR, iV, cR, Y, A) \cup \left(\left((T, S, ID, rY, iR, iV, cR, Y, A) \cap R_2\right)^+ \cap R_2\right) = Oscars = R$$

So, we got that left and right side of the first dependency is in Z so it is preserved, now we check for the third dependency:

$$Z' = (T, D, rY)$$
$$Z = (T, D, rY) \cup \left(\left((T, D, rY) \cap R_1\right)^+ \cap R_1\right) = (T, D, S, ID, rY, iR, iV, cR)$$
So, we can see that we done, because the left and right side in the third dependency belongs to this current Z and it is preserved, now we move to check the fourth dependency:
$$Z' = (T, rY)$$
$$Z = (T, rY) \cup \left(\left((T, rY) \cap R_1\right)^+ \cap R_1\right) = (T, rY, S, ID, iR, iV, cR)$$
$$Z = (T, rY, S, ID, iR, iV, cR) \cup \left(\left((T, rY, S, ID, iR, iV, cR) \cap R_2\right)^+ \cap R_2\right) = (T, rY, S, ID, iR, iV, cR)$$
$$Z = (T, rY, S, ID, iR, iV, cR) \cup \left(\left((T, rY, S, ID, iR, iV, cR) \cap R_3\right)^+ \cap R_3\right) = (T, rY, S, ID, iR, iV, cR, Y, A)$$

So, we can see that we are done with this dependency, and it is preserved.
By all this process we prove that our decomposition is Dependency Preserving, and we done.

Question (2):

$Purchases(TransactionNo, Date, ProductNo, ProductName, Price, Quantity, CustomerNo, Country)$

a) $TransactionNo \rightarrow Date$
$TransactionNo \rightarrow CustomerNo$
$ProductNo \rightarrow ProductName$
$CustomerNo \rightarrow Country$

b) To find the keys of our relation, we need to use the "FindKey" Algorithm we learned in class, we use it with R=Purchases and F=Functional Dependencies from section (a):
We can see that we can remove (Date, ProductName, CustomerNo, Country) from our keys, because they follow from the others, as Date follows from the first FD, ProductName follows from the third FD, CustomerNo follows from the second FD and Country follows from the last FD from section (a), so we got that the Key to our relation is the quartette {TransactionNo, ProductoNo, Price, Quantity}

c) We start to check if the relation is in BCNF:
We can see that not all left sides of our FDs are superkeys, and so for the second condition of BCNF forms, none of our FDs are trivial.
Our relation is not in BCNF, we try to check if it is in 3NF:
All of left sides of our FDs are not superkeys so we go to the second condition of 3NF forms, we can see that also the second condition didn't hold because for example $Date \nsubseteq TranscationNo$, so we got that our relation is not in 3NF.
So, the relation is neither in 3NF nor BCNF.

d) SQL

e) The first two FD holds and the other two don't hold, now we want to find a decomposition, we want to find 3NF decomposition so we use Find3NFDecomposition(R, F) algorithm:

We can see that our FDs is section (a) are minimal cover of our relation, so we continue with the algorithm and we got the following schemas: $R_1 = (TransactionNo, Date), R_2 = (TransactionNo, CustomerNo)$, $R_3 = (ProductNo, ProductName), R_4 = (CustomerNo, Country)$ now we want to add a schema which contains our key so $R_5 = (TransactionNo, ProductNo, Price, Quantity)$, there is no schema that contains in another schema, so we found a 3NF decomposition of R.
With this decomposition we can ensure consistency while inserting data into the table, because as we said before that $ProductNo \rightarrow ProductName$ & $CustomerNo \rightarrow Country$ dependencies have violation, so is this decomposition each one of them alone in a schema, so it will not affect inserting the others.

Question (3):

$R = (A, B, C, D, E)$
$F_R = \{CD \rightarrow AB, A \rightarrow E, E \rightarrow D, ACE \rightarrow B\}$
$R_1 = (A, C, D, E)$
$R_2 = (B, C, D, E)$

a) To find all R's keys, we must use AllKeys algorithm, after using it we found out that R's keys are (AC, CD, CE).

b) We can see that R is not in BCNF, not all left sides in F_R are superkeys and there are no trivial dependencies in it.
Now, we want to check if it is in 3NF:
In the first and last dependencies we can see that left side is a superkey, now we check if the second and third one holds the second condition of being 3NF, in the second one, the right side has "E" in it, it is not in left side but it is in a key, so the second want holds, the third one is the same, to conclude that all the dependencies holds for the conditions for R to be in 3NF, and we done.

c) To check if this decomposition is lossless or not, we create table and then chase it and if we get a row with a_i's we conclude that our decomposition is lossless, the initial table is:

	A	B	C	D	E
ACDE	a_1	$b_{1,2}$	a_3	a_4	a_5
BCDE	$b_{2,1}$	a_2	a_3	a_4	a_5

After chasing the table, we got this table:

	A	B	C	D	E
ACDE	a_1	$b_{1,2}$	a_3	a_4	a_5
BCDE	a_1	a_2	a_3	a_4	a_5

There is no need to continue in the algorithm because we got a row with a's, so we got that our decomposition is lossless, and we are done.

d) We can see that the dependencies 2 and 3 preserved because left and right side belongs to the subschema R_1.
Now we check preserving for dependency 1 and 4, we start with 1:

$Z' = CD$
$Z = CD \cup ((CD \cap R_1)^+ \cap R_1) = ACDE$
$Z = ACDE \cup ((ACDE \cap R_2)^+ \cap R_2) = ACDEB$

We can see that we are done with this dependency, because the left and right side of dependency 1 are in this current Z, and we are done.
Now, we want to check preserving for dependency 4:

$Z' = ACE$
$Z = ACE \cup ((ACE \cap R_1)^+ \cap R_1) = ACED$
$Z = ACED \cup ((ACED \cap R_2)^+ \cap R_2) = ACEDB$

We can see that we are done with this dependency, because the left and right side of dependency 4 are in this current Z, and we are done.

So, we got that our decomposition is Dependency Preserving.

e) $R_1 = (A, C, D, E)$
$F_{R_1} = \{\mathbf{1})A \to (A^+ \cap R_1) = AED,\ \mathbf{2})C \to (C^+ \cap R_1) = C,\ \mathbf{3})D \to (D^+ \cap R_1) = D,$
$\mathbf{4})E \to (E^+ \cap R_1) = ED,\ \mathbf{5})AC \to (AC^+ \cap R_1) = ACED,\ \mathbf{6})AD \to (AD^+ \cap R_1) = ADE,$
$\mathbf{7})AE \to (AE^+ \cap R_1) = AED,\ \mathbf{8})CD \to (CD^+ \cap R_1) = CDAE,\ \mathbf{9})CE \to (CE^+ \cap R_1) = CEDA,$
$\mathbf{10})DE \to (DE^+ \cap R_1) = DE,\ \mathbf{11})ACD \to (ACD^+ \cap R_1) = ACDE,$
$\mathbf{12})\ ACE \to (ACE^+ \cap R_1) = ACED,\ \mathbf{13})ADE \to (ADE^+ \cap R_1) = ADE,$
$\mathbf{14})CDE \to (CDE^+ \cap R_1) = CDEA,\ \mathbf{15})ACDE \to ACDE\}$

To find the minimal cover of this subschema, we use the ComputeMinimalCover algorithm:
Step 1:
$G = \{A \to A, A \to E, A \to D, C \to C, D \to D, E \to E, E \to D, AC \to A, AC \to C, AC \to D, AC \to E,$
$AD \to A, AD \to D, AD \to E, AE \to A, AE \to E, AE \to D, CD \to C, CD \to D, CD \to A, CD \to E,$
$CE \to C, CE \to D, CE \to A, CE \to E, DE \to D, DE \to E, ACD \to A, ACD \to C, ACD \to D, ACD \to E,$
$ACE \to A, ACE \to C, ACE \to D, ACE \to E, ADE \to A, ADE \to D, ADE \to E, CDE \to A, CDE \to C,$
$CDE \to D, CDE \to E, ACDE \to A, ACDE \to C, ACDE \to D, ACDE \to E\}$

Step 2:
$G = \{A \to A, A \to E, A \to D, C \to C, D \to D, E \to E, E \to D, CD \to A, CD \to E, CE \to A$
$A \to A, D \to D, A \to E, A \to A, E \to E, A \to D, C \to C, D \to D, A \to A, C \to C, A \to D, A \to E,$
$C \to C, E \to D, E \to E, D \to D, E \to E, A \to A, C \to C, D \to D, A \to E,$
$A \to A, C \to C, A \to D, E \to E, A \to A, D \to D, E \to E, CD \to A, C \to C,$
$D \to D, E \to E, A \to A, C \to C, D \to D, E \to E\}$

Step 3:
$G = \{A \to E, E \to D, CD \to A, CD \to E\}$

So, we got that the minimal cover for F_{R_1} is $\{A \to E, E \to D, CD \to A, CD \to E\}$.

We can see that R_1 is not in BCNF, because the first dependency didn't hold any condition of BCNF.
So, we try to check if R_1 is in 3NF or not:

We can see that the keys for R1 are {AC, CD, CE}, so the dependencies in F_{R_1} meet the conditions of 3NF, so R1 is in 3NF but not BCNF.

f) $R_2 = (B, C, D, E)$
$F_{R_2} = \{\mathbf{1})B \to (B^+ \cap R_2) = B,\ \mathbf{2})C \to (C^+ \cap R_2) = C,\ \mathbf{3})D \to (D^+ \cap R_2) = D,$
$\mathbf{4})E \to (E^+ \cap R_2) = ED,\ \mathbf{5})BC \to (BC^+ \cap R_2) = BC,\ \mathbf{6})BD \to (BD^+ \cap R_2) = BD,$
$\mathbf{7})BE \to (BE^+ \cap R_2) = BED,\ \mathbf{8})CD \to (CD^+ \cap R_2) = CDBE,\ \mathbf{9})CE \to (CE^+ \cap R_2) = CED,$
$\mathbf{10})DE \to (DE^+ \cap R_2) = DE,\ \mathbf{11})BCD \to (BCD^+ \cap R_2) = BCD,$
$\mathbf{12})\ BCE \to (BCE^+ \cap R_2) = BCED,\ \mathbf{13})BDE \to (BDE^+ \cap R_2) = BDE,$
$\mathbf{14})CDE \to (CDE^+ \cap R_2) = CDEB,\ \mathbf{15})BCDE \to BCDE\}$

Step 1:
$G = \{B \to B, C \to C, D \to D, E \to E, E \to D, BC \to B, BC \to C, BD \to B, BD - D, BE \to B, BE \to E,$
$BE \to D, CD \to C, CD \to D, CD \to B, CD \to E, CE \to C, CE \to E, CE \to D, DE \to D, DE \to E, BCD \to B,$
$BCD \to C, BCD \to D, BCE \to B, BCE \to C, BCE \to E, BCE \to D, BDE \to B, BDE \to D, BDE \to E,$
$CDE \to C, CDE \to D, CDE \to E, CDE \to B, BCDE \to B, BCDE \to C, BCDE \to D, BCDE \to E\}$

Step 2:
$G = \{B \to B, C \to C, D \to D, E \to E, E \to D, CD \to B, CD \to E, B \to B, C \to C, B \to B, D - D, B \to B,$
$E \to E, E \to D, C \to C, D \to D, C \to C, E \to E, E \to D, D \to D, E \to E, B \to B,$
$C \to C, D \to D, B \to B, C \to C, E \to E, E \to D, B \to B, D \to D, E \to E,$
$C \to C, D \to D, E \to E, CD \to B, B \to B, C \to C, D \to D, E \to E\}$

Step 3:
$G = \{E \to D, CD \to B, CD \to E\}$
We got that the minimal cover for R2 is $\{E \to D, CD \to B, CD \to E\}$.

We can see that R2 is not in BCNF because $E \to D$ is not trivial and E is not a superkey, so we try to check if it is in 3NF:
We can see that the conditions of 3NF holds, CD is a key and D is an attribute in the key CD, so we got that R2 is 3NF but not BCNF.

Question (4):

a) The claim is **TRUE**.

Let $Y \to Z$ be a functional dependency that holds over X. We show that $Y \to Z$ satisfies one of the BCNF conditions. There are 2 cases:

a. If $Z = X$ then Y is a super-key ($Y^+ = X$), indeed, it is a key.

b. If $Z \subset X$ then $Z \subseteq Y$, that is, $Y \to Z$ is trivial, otherwise, there is a contradiction to non-redundancy of $X \to A$. Assume towards contradiction that $Z \subset X$ and $Z \nsubseteq Y$, then there are two subsets of attributes $Z, Y \subset X$ such that $Z \nsubseteq Y$ and $Y \to Z$, then $(X - Z)^+ = X^+$, contradiction.

b) The claim is **FALSE**.

Counterexample: let $\mathcal{R} = (A, B, C, D)$ and a minimal cover over $\mathcal{R}$ are $\mathcal{F} = \{AB \to CD\ , C \to B\}$, for the dependency $AB \to C$ we create the subschema $\mathcal{R}_1 = (A, B, C)$ with $\mathcal{F}_{\mathcal{R}_1} = \{AB \to C\ , C \to B\}$, then the dependency $C \to B$ violates the BCNF, as neither C is a super-key ($C^+ = CB \neq \mathcal{R}_1$) nor the dependency is trivial ($B \nsubseteq C$).