ADIL ABUWANI
DATABASE PROJECT 3
CSCI 331-33
MAY 18, 2018
Student-ID: 4504(Last Four Digits)

Adil Abuwani
Adil.abuwani@outlook.com

RELATION 1: CUSTOMER Relation provides all the information about the customer in cvs store

CID	CFIRSTNAME	CLASTNAME	CEMAIL	CPHONE	CSTREET	CCITY	CSTATE	CZIP
1	smith	bob	smith@gmail.com	4578562114	queens blvd	Rego Park	NY	11374
2	smith	jon	smithjon@gmail.com	5478654112	queens blvd	Rego Park	NY	11375
3	tom	jon	tom@gmail.com	5786523558	queens blvd	Rego Park	NY	11379
4	mike	maoura	mile@gmail.com	5478965441	kissena blvd	flushing	NY	11367
5	rod	miky	miky@gmail.com	5478658995	kissena blvd	flushing	NY	11368

CUSTOMER(CID(PK), CFIRSTNAME, CLASTNAME, CEMAIL, CPHONE, CSTREET, CCITY, CSTATE, CZIP)

PRIMARY KEY:

OWNERID

Degree = 9

DOMAINS:

CID(PK): 0-9

CUSTOMERNAME-COMPOSITE ATTRIBUTE THAT CAN BE DIVIDED INTO FURTHER SMALLER COMPONENTS.

CFIRSTNAME: A-Z & A-Z

CLASTNAME: A-Z & A-Z

CEMAIL: A-Z & A-Z & 0-9

CPHONE : 10 DIGITS

CUSTOMERLOCATION-COMPOSITE ATTRIBUTE

CSTREET: A-Z & A-Z

CCITY: A-Z & A-Z

CSTATE: A-Z & A-Z

ZIP: 5 DIGITS

Third Normal form:

We know that a relation is of third normal form if it is in first normal form, second normal form, and in which no non-candidate key is transitively dependent on candidate key.

In the Customer relation, the intersection of each row and column has one and only one value. So, we know that the relation is of first normal form.

Secondly, in the customer relation, the non-primary key is functionally dependent on the primary key.

Finally, we also observe that there exists no non-candidate key which is transitively dependent on candidate key. Hence the relation is of third normal form.

SQL CREATE TABLE:

```
CREATE TABLE CUSTOMER
   (  CID NUMBER,
  CFIRSTNAME VARCHAR(30) NOT NULL,
        CLASTNAME VARCHAR(30) NOT NULL,
        CEMAIL VARCHAR(320) NOT NULL,
        CPHONE CHAR(10) NOT NULL,
```

```
        CSTREET VARCHAR(30) NOT NULL,

        CCity VARCHAR(30) NOT NULL,

        CState VARCHAR(30) NOT NULL,

        CZip CHAR(5) NOT NULL,

  CONSTRAINT "CUSTOMER_PK" PRIMARY KEY (CID)

   );

COMMIT;
```

SQL INSERT VALUES IN TABLE:

```
create sequence SEQ_CUSTOMER
start with 1 increment by 1;
COMMIT;

INSERT INTO CUSTOMER (CID, CFIRSTNAME, CLASTNAME, CEMAIL, CPHONE,
CSTREET, CCity, CState, CZip)

VALUES (SEQ_CUSTOMER.NEXTVAL, 'smith', 'bob', 'smith@gmail.com',
'4578965221', 'queens blvd', 'Rego park', 'NY', '11374')

COMMIT;

INSERT INTO CUSTOMER (CID, CFIRSTNAME, CLASTNAME, CEMAIL, CPHONE,
CSTREET, CCity, CState, CZip)

VALUES (SEQ_CUSTOMER.NEXTVAL, 'smith', 'jon', 'smithjon@gmail.com',
'5547865221' 'queens blvd', 'Rego park', 'NY', '11375')
COMMIT;

INSERT INTO CUSTOMER (CID, CFIRSTNAME, CLASTNAME, CEMAIL, CPHONE,
CSTREET, CCity, CState, CZip)
```

```
VALUES (SEQ_CUSTOMER.NEXTVAL, 'tom', 'jon', 'tom@gmail.com', '5786523558'

'queens blvd', 'Rego park', 'NY', '11379')


COMMIT;
```

(WE WILL INSERT ALL CUSTOMER WITH THE SAME APPROACH,AND MOVE TO THE NEXT RELATION)

RELATION 2: STAFF Relation provides all the information about the staff in CVS store. NOTE THAT each staff is assigned to ONE store.

SID	SFIRSTNAME	SLASTNAME	SEMAIL	SSTOREID	SSALARY	SADDRESS
100	jamil	yakab	jamil@gmail.com	50	400	kissena blvd
101	tom	superman	tom@gmail.com	55	400	queens blvd
102	rex	genrun	rex@gmail.com	52	400.5	broadway
103	toz	gen	toz@gmail.com	53	500.5	queens blvd
104	rim	rins	rim@gmail.com	54	700.5	queens blvd

STAFF(SID(PK), SFIRSTNAME, SLASTNAME, SEMAIL, SSTOREID, SSALARY, SADDRESS)

PRIMARY KEY:

SID

Degree = 7

FOREIGN KEY:

SSID, references STORE(STOREID)

DOMAINS:

SID(PK): 0-9

STAFFNAME-COMPOSITE ATTRIBUTE THAT CAN BE DIVIDED INTO FURTHER SMALLER COMPONENTS.

SFIRSTNAME: A-Z & A-Z

SLASTNAME: A-Z & A-Z

SEMAIL: A-Z & A-Z & 0-9

SPHONE : 10 DIGITS

SSTOREID: 0-9 & ALL THE VALID STORE ID FROM STORE RELATION

SSALARY: 0-9 IN US DOLLAR.

SQL CREATE TABLE:

```
CREATE TABLE STAFF
   (  SID NUMBER,
  SFIRSTNAME VARCHAR(30) NOT NULL,
        SLASTNAME VARCHAR(30) NOT NULL,
        SEMAIL VARCHAR(320) NOT NULL,
        SSTOREID NUMBER NOT NULL,
        SSALARY NUMBER(10, 2) NOT NULL,
        SADDRESS VARCHAR(50) NOT NULL,
  CONSTRAINT "STAFF_PK" PRIMARY KEY (SID)
   );
```

COMMIT;

SQL FORIGN KEYS:

ALTER TABLE STAFF

ADD FOREIGN KEY (SSTOREID) REFERENCES CVSSTORE(STOREID);

SQL INSERT VALUES IN TABLE:

```
create sequence SEQ_STAFF

start with 100 increment by 1;


INSERT INTO STAFF VALUES (SEQ_STAFF.NEXTVAL, 'jamil', 'yakab',
'jamil@gmail.com', 50, 400, 'kissena blvd');

COMMIT;

INSERT INTO STAFF VALUES (SEQ_STAFF.NEXTVAL, 'tom', 'superman',
'tom@gmail.com', 51, 400, 'queens blvd');

COMMIT;

INSERT INTO STAFF VALUES (SEQ_STAFF.NEXTVAL, 'rex', 'genrun',
'rex@gmail.com', 52, 400.50, 'broadway');

COMMIT;

INSERT INTO STAFF VALUES (SEQ_STAFF.NEXTVAL, 'toz', 'gen',
'toz@gmail.com', 53, 500.50, 'queens blvd');

COMMIT;

INSERT INTO STAFF VALUES (SEQ_STAFF.NEXTVAL, 'rim', 'rins',
'rim@gmail.com', 54, 700.50, 'queens blvd');

COMMIT;
```

RELATION 3: CVSSTORE Relation provides all the information about the CVS store located.

STOREID	STORESTREET	STORECITY	STORESTATE	STOREZIP
50	queens blvd	Rego Park	NY	11375
51	queens blvd	Rego Park	NY	11376
52	kissena blvd	flushing	NY	11367
53	broadway	queens blvd	NY	11348
54	thomson ave	long island city	NY	11101
55	broadway	queens blvd	NY	11399

CVSSTORE(STOREID(PK), STORESTREET, STORECITY, STORESTATE, STOREZIP)

PRIMARY KEY:

STOREID

Degree = 5

DOMAINS:

STOREID(PK): 0-9

STORELOCATION-COMPOSITE ATTRIBUTE

STORESTREET: A-Z & A-Z

STORECITY: A-Z & A-Z

STORESTATE: A-Z & A-Z

STOREZIP: 5 DIGITS

SQL CREATE TABLE:

```
CREATE TABLE CVSSTORE
   (  STOREID NUMBER,
  STORESTREET VARCHAR(30) NOT NULL,
        STORECITY VARCHAR(30) NOT NULL,
        STORESTATE VARCHAR(30) NOT NULL,
        STOREZIP CHAR(5) NOT NULL,
  CONSTRAINT "STORE_PK" PRIMARY KEY (STOREID)
   );
COMMIT;
```

SQL INSERT VALUES IN TABLE:

```
create sequence SEQ_CVSSTORE
```

```
start with 50 increment by 1;

COMMIT;


INSERT INTO CVSSTORE (STOREID, STORESTREET, STORECITY, STORESTATE,
STOREZIP)

VALUES (SEQ_CVSSTORE.NEXTVAL, 'queens blvd', 'Rego Park', 'NY', '11375');

COMMIT;

INSERT INTO CVSSTORE (STOREID, STORESTREET, STORECITY, STORESTATE,
STOREZIP)

VALUES(SEQ_CVSSTORE.NEXTVAL, 'queens blvd', 'Rego Park', 'NY', '11376');

COMMIT;

INSERT INTO CVSSTORE (STOREID, STORESTREET, STORECITY, STORESTATE,
STOREZIP)

VALUES(SEQ_CVSSTORE.NEXTVAL, 'kissena blvd', 'flushing', 'NY', '11367');

COMMIT;

INSERT INTO CVSSTORE (STOREID, STORESTREET, STORECITY, STORESTATE,
STOREZIP)

VALUES(SEQ_CVSSTORE.NEXTVAL, 'thompson ave', 'long island city', 'NY',
'11101');

COMMIT;

(WE WILL INSERT ALL CVS STORE WITH THE SAME APPROACH,AND MOVE TO THE NEXT
RELATION)
```

RELATION 4: DRUG Relation provides all the information about the drug list in all CVS store.

DRUG(DID(PK), DNAME)

PRIMARY KEY:

DID

Degree =2

DOMAINS:

DID(PK): 0-9

DNAME= A-Z & A-Z

SQL CREATE TABLE:

```
CREATE TABLE DRUG
   (  DID NUMBER,
  DNAME VARCHAR(50) NOT NULL,
  CONSTRAINT "DRUG_PK" PRIMARY KEY (DID)
   );
COMMIT;
```

SQL INSERT VALUES IN TABLE:

```
create sequence SEQ_DRUG
start with 200 increment by 1;
COMMIT;
INSERT INTO DRUG VALUES(SEQ_DRUG.NEXTVAL, 'ADVIL')
COMMIT;
INSERT INTO DRUG VALUES(SEQ_DRUG.NEXTVAL, 'TYLENOL')
COMMIT;
INSERT INTO DRUG VALUES(SEQ_DRUG.NEXTVAL, 'PEOTOBESIMOL')
```

```
COMMIT;

INSERT INTO DRUG VALUES(SEQ_DRUG.NEXTVAL, 'ENERGYC')

COMMIT;

INSERT INTO DRUG VALUES(SEQ_DRUG.NEXTVAL, 'PANADOL')

COMMIT;

INSERT INTO DRUG VALUES(SEQ_DRUG.NEXTVAL, 'ZENTEX')

COMMIT;

INSERT INTO DRUG VALUES(SEQ_DRUG.NEXTVAL, 'SAMURIX')

COMMIT;
```

DID	DNAME
200	ADVIL
201	PEOTOBESIMOL
202	ENERGYC
203	PANADOL
204	ZENTEX
205	SAMURIX

Since the drug price is different at each CVS store, to prevent from repetition, we will create a new relation that keeps track of each price of the drug at a specific store.

RELATION 5: DRUGSTOREDETAIL Relation provides all the information about the price list of each drug in a specific CVS store.

DID	DSTOREID	DPRICE	DSTATUS
200	50	2	A
201	50	2	A
202	50	3	A
203	50	3	A
204	50	3	A
200	51	3	A
201	51	3	A
202	51	1	A
203	51	1	A
200	52	1	A
201	52	1	A
202	52	1	A
200	53	1	A
201	53	1	A
200	54	2	A
More than 15 rows available. Increase rows selector to view more rows.			

DRUGSTOREDETAIL (DID, DSTOREID, DPRICE, DSTATUS)

PRIMARY KEY:

Composite Key: DID(FK), DSTOREID(FK), DPRICE can form a unique key for their occurrence as a **primary Key**.

Degree =4

DOMAINS:

DID(FK): 0-9 & ALL THE VALID DRUG ID IN DRUG RELATION

DSTOREID(FK): 0-9 & ALL THE VALID STORE ID FROM STORE RELATION

DPRICE: 0-9 in US dollar $

DSTATUS: A or N

SQL CREATE TABLE:

```
CREATE TABLE DRUGSTOREDETAIL

    (  DID NUMBER,
```

```
  DSTOREID NUMBER NOT NULL,
        DPRICE NUMBER NOT NULL,
        DSTATUS VARCHAR(1),
  CONSTRAINT "DRUGSTOREDETAIL_PK" PRIMARY KEY (DID, DSTOREID, DPRICE)
   );
```

COMMIT;

SQL FORIGN KEYS:

ALTER TABLE DRUGSTOREDETAIL

ADD FOREIGN KEY (DSTOREID) REFERENCES CVSSTORE(STOREID);

COMMIT;

ALTER TABLE DRUGSTOREDETAIL

ADD FOREIGN KEY (DID) REFERENCES DRUG(DID);

COMMIT;

SQL INSERT VALUES IN TABLE:

```
create sequence SEQ_DRUGSTOREDETAIL
start with 300 increment by 1;
COMMIT;
INSERT INTO DRUGSTOREDETAIL VALUES(200, 50, 2, 'A');
COMMIT;
```

```
INSERT INTO DRUGSTOREDETAIL VALUES(201, 50, 2, 'A');

COMMIT;

INSERT INTO DRUGSTOREDETAIL VALUES(202, 50, 3, 'A');

COMMIT;

INSERT INTO DRUGSTOREDETAIL VALUES(203, 50, 3, 'A');

COMMIT;

INSERT INTO DRUGSTOREDETAIL VALUES(204, 50, 3, 'A');

COMMIT;

INSERT INTO DRUGSTOREDETAIL VALUES(200, 51, 3, 'A');

COMMIT;

INSERT INTO DRUGSTOREDETAIL VALUES(201, 51, 3, 'A');

COMMIT;

INSERT INTO DRUGSTOREDETAIL VALUES(202, 51, 1, 'A');

COMMIT;

INSERT INTO DRUGSTOREDETAIL VALUES(203, 51, 1, 'A');

COMMIT;

INSERT INTO DRUGSTOREDETAIL VALUES(200, 52, 1, 'A');

COMMIT;

INSERT INTO DRUGSTOREDETAIL VALUES(201, 52, 1, 'A');

COMMIT;

INSERT INTO DRUGSTOREDETAIL VALUES(202, 52, 1, 'A');

COMMIT;

INSERT INTO DRUGSTOREDETAIL VALUES(200, 53, 1, 'A');

COMMIT;

INSERT INTO DRUGSTOREDETAIL VALUES(201, 53, 1, 'A');
```

```
COMMIT;

INSERT INTO DRUGSTOREDETAIL VALUES(200, 54, 2, 'A');

COMMIT;

INSERT INTO DRUGSTOREDETAIL VALUES(201, 54, 2, 'A');

COMMIT;

INSERT INTO DRUGSTOREDETAIL VALUES(201, 55, 2, 'A');

COMMIT;
```

RELATION 6: TRANSACTION Relation provides all the transaction information of the customer.

TRANSACTION (ORDERID, CID, DID, STOREID, DOP, PRICE)

Degree =6

DOMAINS:

ORDERID(PK): 0-9

CID(FK): 0-9 & ALL THE VALID CUSTOMER ID FROM CUSTOMER RELATION

DID: 0-9 & ALL THE DRUGID IN DRUG RELATION

STOREID: 0-9 & ALL THE STORE ID IN STORE RELATION

DOP: MM/DD/YYYY

PRICE: 0-9 in US dollar $

```
SQL CREATE TABLE:

Create table transaction

(orderid number primary key,
```

```
cid number not null,

did number not null,

storeid number not null,

dop date not null,

price number(10,2) not null

);
```

SQL FORIGN TABLE:

```
ALTER TABLE transaction
ADD FOREIGN KEY (STOREID) REFERENCES CVSSTORE(STOREID);
COMMIT;


ALTER TABLE transaction
ADD FOREIGN KEY (DID) REFERENCES DRUG(DID);
COMMIT;


ALTER TABLE transaction
ADD FOREIGN KEY (CID) REFERENCES CUSTOMER(CID);
COMMIT;
```

SQL INSERT VALUES IN TABLE:

```
create sequence SEQ_TRANSACTION

start with 500 increment by 1;

INSERT INTO TRANSACTION VALUES(SEQ_TRANSACTION.NEXTVAL, 1, 200, 50,
'3/3/2018', 2 )

COMMIT;

INSERT INTO TRANSACTION VALUES(SEQ_TRANSACTION.NEXTVAL, 2, 201, 50,
'3/3/2018', 2 )

COMMIT;

INSERT INTO TRANSACTION VALUES(SEQ_TRANSACTION.NEXTVAL, 3, 202, 50,
```

```
'3/3/2018', 2 )

COMMIT;

INSERT INTO TRANSACTION VALUES(SEQ_TRANSACTION.NEXTVAL, 1, 201, 50,
'3/3/2017', 2 )

COMMIT;

INSERT INTO TRANSACTION VALUES(SEQ_TRANSACTION.NEXTVAL, 2, 201, 51,
'3/3/2017', 3 )

COMMIT;

INSERT INTO TRANSACTION VALUES(SEQ_TRANSACTION.NEXTVAL, 3, 200, 52,
'3/5/2017', 3 )

COMMIT;

INSERT INTO TRANSACTION VALUES(SEQ_TRANSACTION.NEXTVAL, 1, 203, 53,
'3/6/2017', 3 )

COMMIT;

INSERT INTO TRANSACTION VALUES(SEQ_TRANSACTION.NEXTVAL, 4, 200, 54,
'3/8/2017', 3 )

COMMIT;
```

QUESTIONS:

1. Increase the price of **Advil** by**20%** for only stores in NY. Identify the SQL required to implement. Display the price before and after the price increase.

```
UPDATE DRUGSTOREDETAIL
SET
DRUGSTOREDETAIL.DPRICE=DRUGSTOREDETAIL.DPRICE+(DRUGSTOREDETAIL.DPRICE*0.2
)
WHERE DRUGSTOREDETAIL.DID IN (
SELECT DRUGSTOREDETAIL.DID
FROM DRUGSTOREDETAIL, DRUG, CVSSTORE
WHERE DRUGSTOREDETAIL.DSTOREID=CVSSTORE.STOREID  AND
      CVSSTORE.STORESTATE LIKE '%NY%' AND
      DRUG.DID=DRUGSTOREDETAIL.DID AND
     UPPER( DRUG.DNAME) LIKE '%ADVIL%'

);
COMMIT;
```

BEFORE:

STORESTATE	DNAME	DID	PRICE
NY	ADVIL	200	$2.00
NY	ADVIL	200	$3.00
NY	ADVIL	200	$1.00
NY	ADVIL	200	$1.00
NY	ADVIL	200	$2.00

AFTER:

PRICE
$2.40
$3.60
$1.20
$1.20
$2.40

5 rows retu

2. Identify drugs that have not been purchased **this month**. Display the drug name. Use a nested select to answer this question.

Let this month be the date : 4/19/2018

```
SELECT DRUG.DNAME "DRUG NOT PURCHASED"
FROM DRUG
WHERE DRUG.DID NOT IN (
SELECT TRANSACTION.DID
FROM TRANSACTION
WHERE TO_DATE(TRANSACTION.DOP, 'MM/DD/YYYY')>='4/19/2018'

);
COMMIT;
```

DRUG NOT PURCHASED
ADVIL
PEOTOBESIMOL
ENERGYC
PANADOL
ZENTEX
SAMURIX

3. Reassign employee **tom superman** from the **queens blvd** store to the **broadway** store. Identify the SQL required to implement.

```
UPDATE STAFF
SET STAFF.SSTOREID = (

SELECT CVSSTORE.STOREID
FROM CVSSTORE
WHERE CVSSTORE.STORESTREET='broadway' AND CVSSTORE.STOREZIP='11399'
)
WHERE STAFF.SFIRSTNAME ='tom' AND STAFF.SLASTNAME='superman'
```

SID	SFIRSTNAME	SLASTNAME	SEMAIL	SSTOREID	SSALARY	STOREID	STORESTREET	STORECITY	STORESTATE	STOREZIP
101	tom	superman	tom@gmail.com	55	400	55	broadway	queens blvd	NY	11399

4 rows returned in 0.00 seconds

4. The drug **Advil** will no longer be sold at any store. Identify the SQL required to implement.

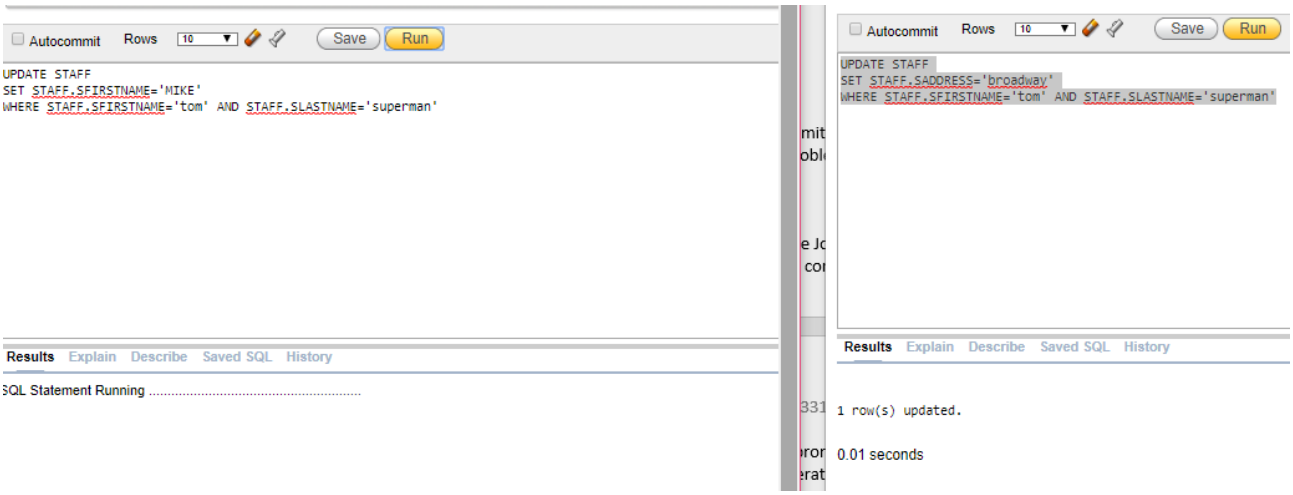
```
UPDATE DRUGSTOREDETAIL
SET DRUGSTOREDETAIL.DSTATUS='N'

WHERE DRUGSTOREDETAIL.DID=(
SELECT DRUG.DID
FROM DRUG
WHERE DRUG.DNAME LIKE '%ADVIL%'

);

COMMIT;
```

DSTATUS	DNAME	DSTOREID
N	ADVIL	50
A	PEOTOBESIMOL	50
A	ENERGYC	50
A	PANADOL	50
A	ZENTEX	50
N	ADVIL	51
A	PEOTOBESIMOL	51
A	ENERGYC	51
A	PANADOL	51
N	ADVIL	52
A	PEOTOBESIMOL	52
A	ENERGYC	52
N	ADVIL	53
A	PEOTOBESIMOL	53
N	ADVIL	54

More than 15 rows available. Increase rows selector to view more rows.

5. In one SQL window, change the address of CVS employee **tom superman**. Don't commit. In another SQL window, change the first name for CVS employee **tom superman**. Don't commit. Explain your results. Resolve the problem. Disable the auto commit flag at the top of the window before performing this operation.

Screen 2: Screen 1:

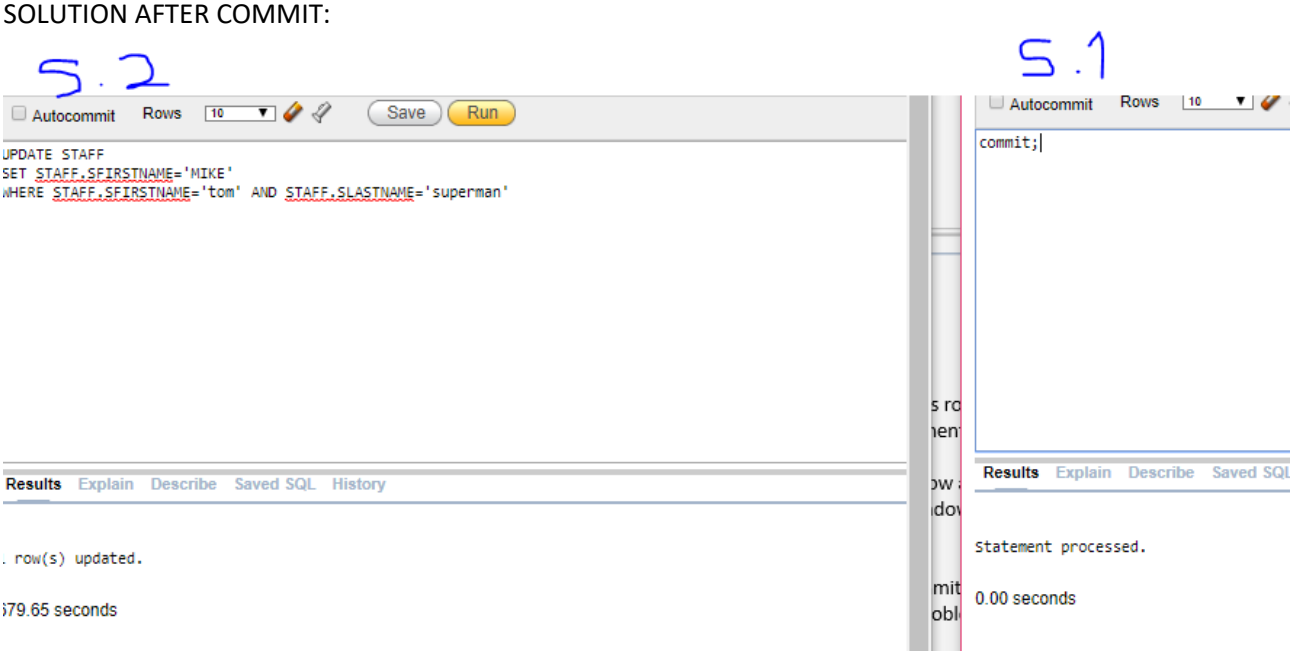


While one SQL window on the right hand side says rows updated, the other one on the left hand side isn't moving, in other words the screen 2 on the left hand says "SQL Statement running" or simply waiting until the commit from the first SQL window. The message never went.

This happens only if both are updating the same row associated with tom superman as it put a lock on the row.

The solution was that, after I commit the first SQL window on the right, the other SQL has started moving again, and the row was updated allowing me to enter commands.

SOLUTION AFTER COMMIT:



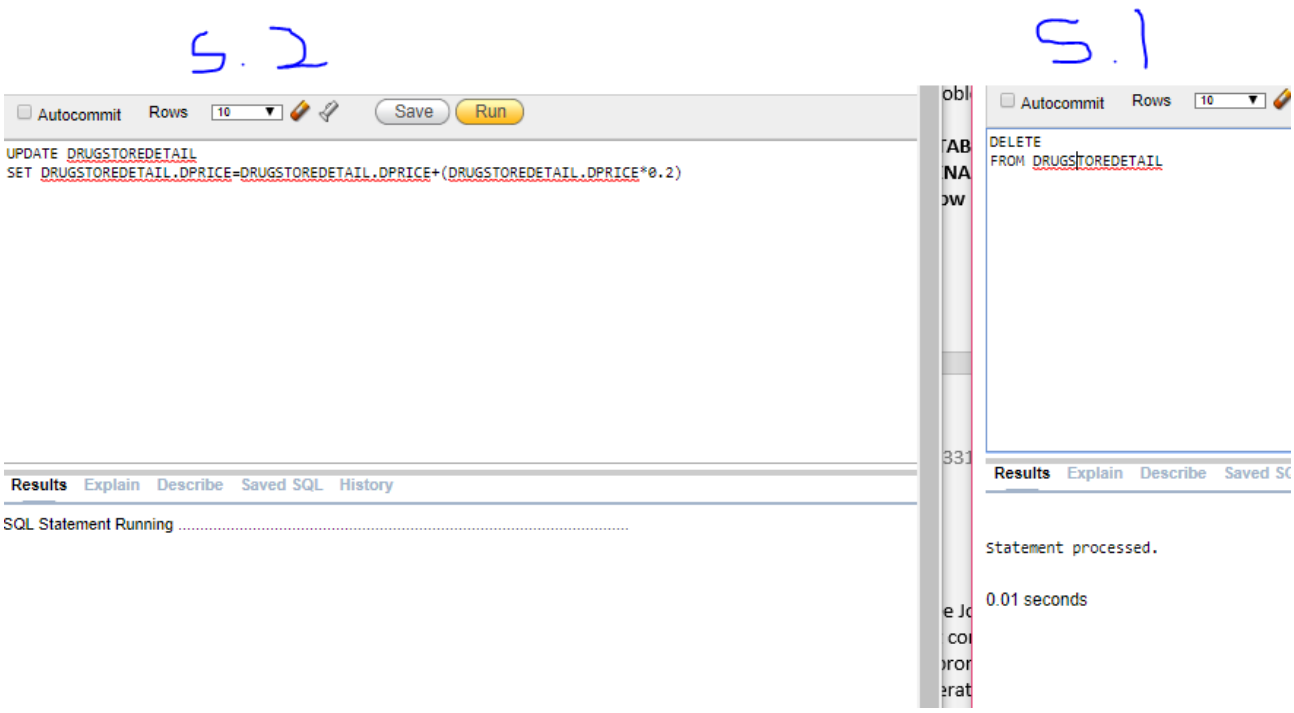
6. In one SQL window, delete all products. Don't commit. In another SQL window increase the price of all products by **20**%. Explain your results. Resolve the problem.

TABLE <NEWTABLE> AS SELECT * FROM <ORIGINALTABLE>;
COMMIT; Then you can rename a table using the RENAME TABLE commit.
Disable the auto commit flag at the top of the window before performing this operation.

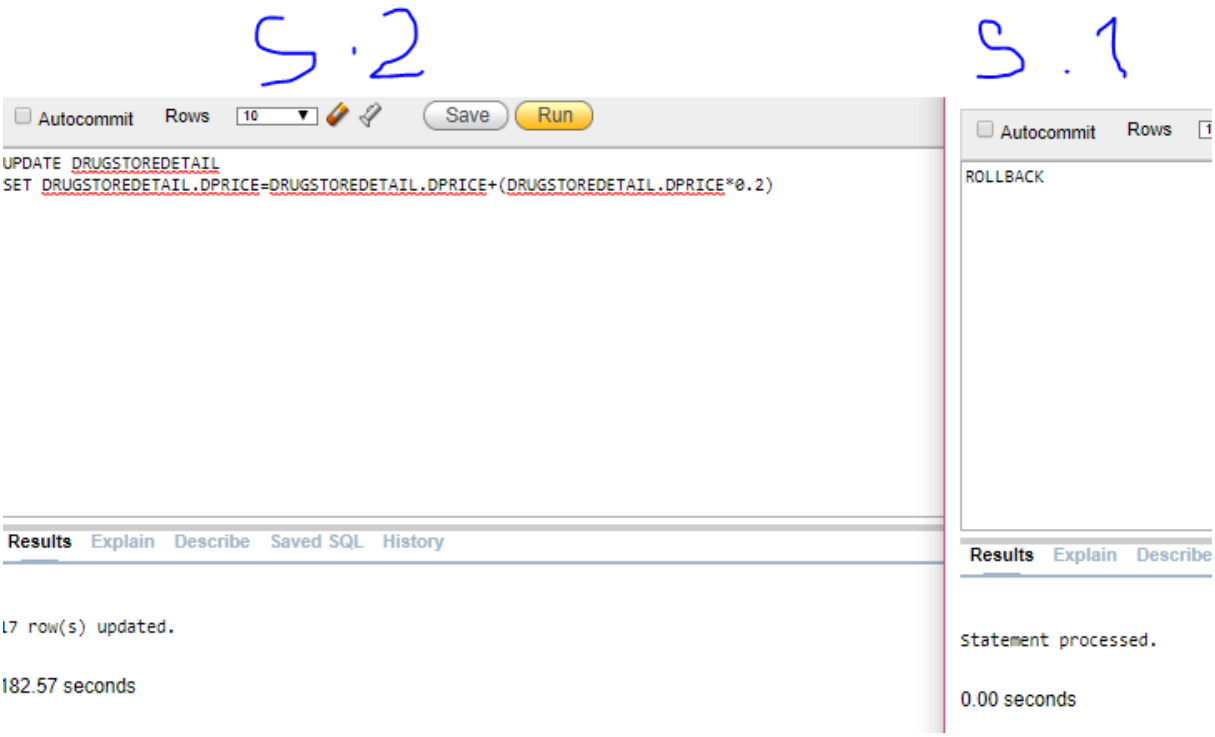
KEEPING THIS IN MIND, I WILL FIRST CREATE A BACKUP TABLE FOR MY DATA SAFETY.

```
CREATE TABLE DRUGSTOREDETAIL2
AS SELECT * FROM DRUGSTOREDETAIL;
COMMIT;
```

THEN PROCEEDED TO THE COMMAND:



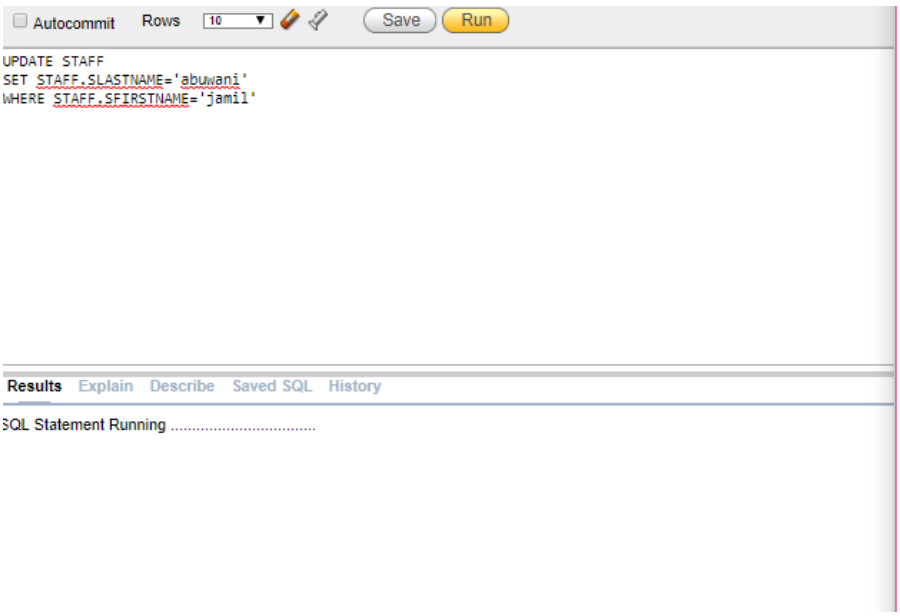
SOLUTION AFTER ROLLBACK:

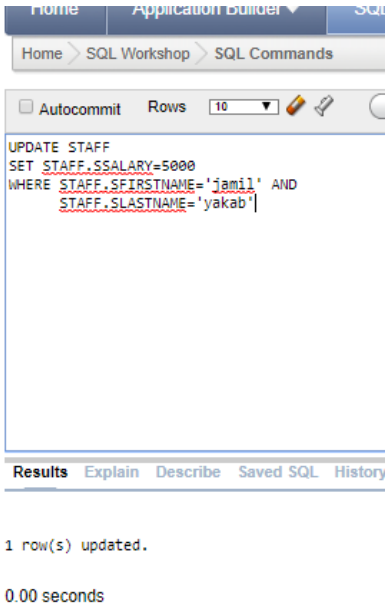


As I ran the first ran the delete statement to delete all the products, it asserted that the “statement processed” however, on the second window, when I ran the update command, I got the same message again indicating SQL Statement running” . The message never went. This problem occurred because all the rows in the table DRUGSTOREDETAIL got locked after window called the delete statement, and did not call the commit. Therefore, the second window could not perform any changes on those rows.

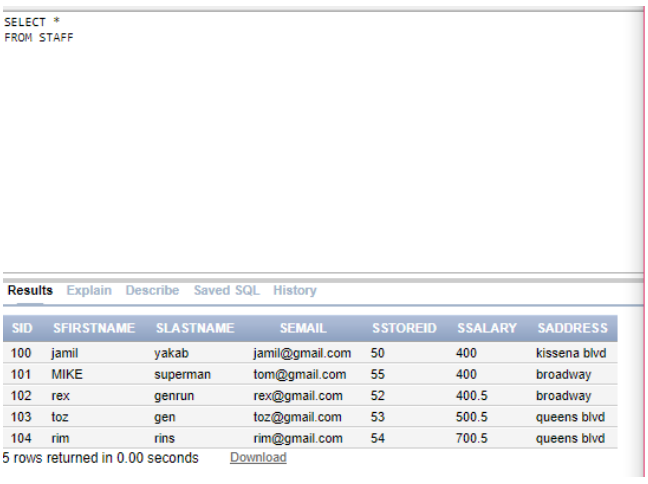
The solution was that I ran the rollback command on screen 1, and as soon as I ran the rollback statement it processed and the update on screen 2 also came to effect. Thus, I the moved further and used the COMMIT command on screen 2 in order for the changes to occur on the DRUGSTOREDETAIL.

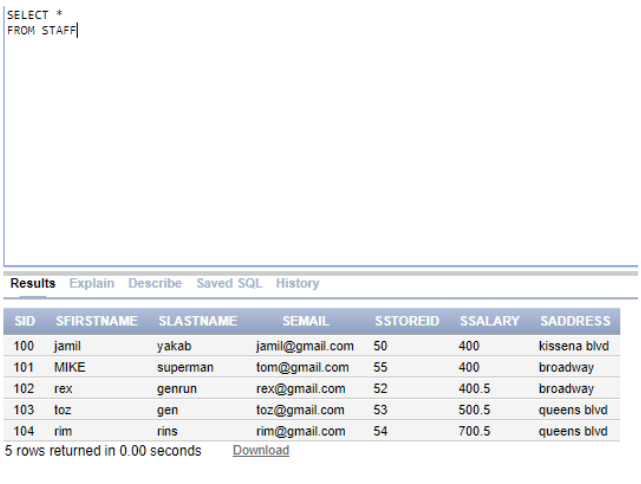
7. In one SQL window, change the salary for employee Jose Lebron. Don't commit. In another SQL window, change the last name of employee Jose Lebron Don’t commit. Quit both Oracle sessions. Login to Oracle and display all information for the employee Jose Lebron. Explain your results. Disable the auto commit flag at the top of the windows before performing this operation.





Now after quitting both oracle sessions, the result obtained is:
We observe that the result did not change because I did not commit the changes for any of the statement. In order for us to get the changes, we need to commit after executing the statement.





8. Use the SQL DESCRIBE operation to display the structure for all tables.

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
CUSTOMER	CID	NUMBER	22	-	-	1	-	-	-
	CFIRSTNAME	VARCHAR2	30	-	-	-	-	-	-
	CLASTNAME	VARCHAR2	30	-	-	-	-	-	-
	CEMAIL	VARCHAR2	320	-	-	-	-	-	-
	CPHONE	CHAR	10	-	-	-	-	-	-
	CSTREET	VARCHAR2	30	-	-	-	-	-	-
	CCITY	VARCHAR2	30	-	-	-	-	-	-
	CSTATE	VARCHAR2	30	-	-	-	-	-	-
	CZIP	CHAR	5	-	-	-	-	-	-
									1 - 9

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
STAFF	SID	NUMBER	22	-	-	1	-	-	-
	SFIRSTNAME	VARCHAR2	30	-	-	-	-	-	-
	SLASTNAME	VARCHAR2	30	-	-	-	-	-	-
	SEMAIL	VARCHAR2	320	-	-	-	-	-	-
	SSTOREID	NUMBER	22	-	-	-	-	-	-
	SSALARY	NUMBER	-	10	2	-	-	-	-
	SADDRESS	VARCHAR2	50	-	-	-	✓	-	-
									1 - 7

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
CVSSTORE	STOREID	NUMBER	22	-	-	1	-	-	-
	STORESTREET	VARCHAR2	30	-	-	-	-	-	-
	STORECITY	VARCHAR2	30	-	-	-	-	-	-
	STORESTATE	VARCHAR2	30	-	-	-	-	-	-
	STOREZIP	CHAR	5	-	-	-	-	-	-
									1 - 5

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
DRUG	DID	NUMBER	22	-	-	1	-	-	-
	DNAME	VARCHAR2	50	-	-	-	-	-	-
									1 - 2

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
DRUGSTOREDETAIL	DID	NUMBER	22	-	-	1	-	-	-
	DSTOREID	NUMBER	22	-	-	2	-	-	-
	DPRICE	NUMBER	22	-	-	3	-	-	-
	DSTATUS	VARCHAR2	1	-	-	-	✓	-	-
									1 - 4

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
TRANSACTION	ORDERID	NUMBER	22	-	-	1	-	-	-
	CID	NUMBER	22	-	-	-	-	-	-
	DID	NUMBER	22	-	-	-	-	-	-
	STOREID	NUMBER	22	-	-	-	-	-	-
	DOP	DATE	7	-	-	-	-	-	-
	PRICE	NUMBER	-	10	2	-	-	-	-
									1 - 6

9. Display the Oracle version by entering select * from product_component_version;

PRODUCT	VERSION	STATUS
NLSRTL	11.2.0.2.0	Production
Oracle Database 11g Express Edition	11.2.0.2.0	64bit Production
PL/SQL	11.2.0.2.0	Production
TNS for 64-bit Windows:	11.2.0.2.0	Production

4 rows returned in 0.08 seconds Download

EXTRA CREDIT QUESTION:

10. Utilize Oracle SQL security roles to limit access to view customer information. This security role will only be able to display and not add, delete or change customer information. Identify the SQL operations to implement and demonstrate the functionality of the security roles

In order to limit access to a user so that the user can only view information of the customer without granting permission to update, insert or delete. Since we are assigned the security role, we have the privilege to limit access to the users. In other words we can grant the users certain privillages.

To do so, we will first create a user that is identified by the administrator.

COMMAND:

```
create user <username >identified by <password>
```

```
CREATE USER sam IDENTIFIED BY SEC;
```

Now, we will grant the user limited privileges such as to login and create a session through:

```
GRANT CONNECT TO sam;
```

COMMAND:

```
GRANT CREATE SESSION TO <username>
```

```
GRANT CREATE SESSION TO sam;
```

```
GRANT SELECT ON CUSTOMER TO sam;
```

```
Therefore, the user now has the ability to view the object CUSTOMER with
the privilege to view data in the customer.
```

11. Utilize Oracle SQL security roles to allow users to purchase a product at a CVS store, but prevent the deletion or changes to transactions. Identify the SQL operations to implement and demonstrate the functionality of the security roles.

To utilize this security role to allow users to purchase a product at a cvs store,and prevent them from deleting/making updates. I will first follow the first two steps from the previous question we will first create a user that is identified by the administrator.

Create user:

```
CREATE USER sam IDENTIFIED BY SEC;
```

GRANT USER TO LOGIN AND CREATE A SESISON:

```
GRANT CONNECT TO sam;

GRANT CREATE SESSION TO sam;
```

GRANT PRIVILLAGE TO ALLOW USERS TO PURCHASE:

```
GRANT INSERT ON TRANSACTION TO sam;
```