Name- Aarya Sanjay Dange

Assignment – 17

Constraining the Values of your data.

1) Create the Orders table so that all onum values as well as all combinations of cnum and snum are different from one another, and so that NULL values are excluded from the date field.

```
mysql> alter table orders
    -> add constraint uniq_onum unique (onum);
mysql> ALTER TABLE orders
    -> ADD CONSTRAINT uniq_cnum_snum UNIQUE (cnum, snum);
mysql> alter table orders
    -> modify odate date not null;
```

2) Create the Salespeople table so that the default commission is 10% with no NULLS permitted, snum is the primary key, and all names fall alphabetically between A and M, inclusive (assume all names will be uppercase).

```
mysql> Create table salespeople2
        (snum int primary key,
        Sname varchar(50) not null check (sname between 'A' and 'M'  ),
        city varchar(50),
        comm decimal(3,2) not null default 0.10 check (comm between 0.10
        and 1.00));
```

3) Create Orders table,making sure that onum is greater than cnum, and cnum is greater that snum. Allow no NULLS in any of these three fields.

```
mysql> create table orders2
    -> (snum int primary key,  cnum int not null,
    -> onum int not null,
    -> check (cnum > snum), check (onum > cnum));
```

Assignment – 18

Maintaining the Integrity of your Data.

1) Create a table called Cityorders. This will contain the same onum, amt and snum fields as the Orders table, and the same cnum and city fields as the Customers table, so that each customer's order will be entered into this table along with his or her city. Onum will be the primary key of Cityorders. All of the fields in Cityorders will be constrained to match the Customers and Orders tables. Assume the parent keys in these tables already have the proper constraints.

```
mysql> Create table Cityorders(
    -> Onum int primary key,
    -> Amt decimal(10,2),
    -> Snum int,
    -> Cnum int,
    -> City varchar(50)
    -> );
Query OK, 0 rows affected (0.05 sec)

mysql> INSERT INTO Cityorders (Onum, Amt, Snum, Cnum, City)
    -> SELECT orders.Onum, orders.Amt, orders.Snum, customers.Cnum,
customers.City
    -> FROM Orders,customers
    -> Where  orders.Cnum = customers.Cnum;
Query OK, 6 rows affected, 5 warnings (0.02 sec)
Records: 6  Duplicates: 0  Warnings: 5
```

2) Redefine the Orders table as follows:- add a new column called prev, which will identify, for each order, the onum of the previous order for that current customer. Implement this with a foreign key referring to the Orders table itself. The foreign key should refer as well to the cnum of the customer, providing a definite enforced link between the current order and the one referenced.

```
Alter table orders

Add column prev int,

Add constraint fk_prev_orders foreign key (prev) references orders (onum);

Update orders o1 set prev = (

Set prev = (

Select max(o2.onum)  from orders o2

Where o2.cnum = o1.cnum and o2.onum<o1.onum

);
```

Assignment – 19

Views.

1) Create a view that shows all of the customers who have the highest ratings.

```
mysql> create view v1 as
   -> select * from customers where rating = (
   -> select max(rating) from customers);
Query OK, 0 rows affected (0.03 sec)

mysql> select * from v1;
```

2) Create a view that shows the number of salespeople in each city.

```
mysql> create view v2 as
      -> select city,count(sname) from salespeople group by city;
Query OK, 0 rows affected (0.01 sec)

mysql> select * from v2;
```

3) Create a view that shows the average and total orders for each salesperson after his or her name. Assume all names are unique.

```
mysql> create view v3 as
   -> select sname,avg(amt),sum(amt) from orders,salespeople where
salespeople.snum=orders.snum group by orders.snum;
Query OK, 0 rows affected (0.02 sec)

mysql> select * from v3;
```

4) Create a view that shows each salesperson with multiple customers.

```
mysql> Create view v4 as select
salespeople.sname,salespeople.snum,count(customers.cnum) from
customers,salespeople where customers.snum =salespeople.snum group
by salespeople.snum,salespeople.sname having count(cnum)>1;
Query OK, 0 rows affected (0.02 sec)

mysql> select * from v4;
```

Assignment – 20
Changing Values through Views

1) Which of these views are updateable (will allow DML operations)?

#1 Create View Dailyorders
as Select Distinct cnum, snum, onum, odate from Orders;

```
 select * from Dailyorders;

+------+------+------+------------+

| cnum | snum | onum | odate      |

+------+------+------+------------+

| 2008 | 1007 | 3001 | 1990-10-03 |

| 2001 | 1001 | 3003 | 1990-10-03 |

| 2007 | 1004 | 3002 | 1990-10-03 |

| 2003 | 1002 | 3005 | 1990-10-03 |

| 2008 | 1007 | 3006 | 1990-10-03 |

| 2002 | 1003 | 3009 | 1990-10-04 |

| 2004 | 1002 | 3007 | 1990-10-04 |
| 2006 | 1001 | 3008 | 1990-10-05 |

| 2004 | 1002 | 3010 | 1990-10-06 |

| 2006 | 1001 | 3011 | 1990-10-06 |

+------+------+------+------------+
```

#2   Create View Custotals
        as Select cname, Sum (amt) Sum_Amt from Orders, Customers
        where Orders.cnum=Customers.cnum
        Group by cname;

```
select * from Custotals;

+----------+----------+

| cname      | Sum_Amt  |

+----------+----------+

| Cisneros |  1116.85 |

| Hoffman  |   767.19 |

| Pereira  |  1900.10 |

| Liv   |  5160.45 |

| Giovanni |  1713.23 |

| Grass       |  1385.70 |

| Clemens  | 14614.88 |

+----------+----------+
```

#3   Create view Thirdorders
        as Select * from Dailyorders
        where odate='1990-10-03';

```
select * from Thirdorders;

+------+------+------+------------+

| cnum | snum | onum | odate      |

+------+------+------+------------+

| 2008 | 1007 | 3001 | 1990-10-03 |

| 2001 | 1001 | 3003 | 1990-10-03 |

| 2007 | 1004 | 3002 | 1990-10-03 |

| 2003 | 1002 | 3005 | 1990-10-03 |

| 2008 | 1007 | 3006 | 1990-10-03 |

+------+------+------+------------+
```

```
#4   Create view Nullcities
      as Select snum, sname, city
      from Salespeople
      where city is NULL
      OR sname BETWEEN 'A' and 'MZ';
```

```
select * from Nullcities;

+------+---------+----------+

| snum | sname   | city     |

+------+---------+----------+

| 1004 | Motika  | London   |

| 1003 | Axelrod | new   |

| 1100 | Blanco  | san jose |
```

2) Create a view of the Salespeople table called Commissions. This view will include only the snum and comm fields. Through this view, someone could enter or change commissions, but only to values between .10 and .20.

```
CREATE VIEW Commissions AS

SELECT snum, comm

FROM Salespeople;
```

3) Some SQL implementations have a built-in constant representing the current date, sometimes called “CURDATE” or “SYSDATE”. The word SYSDATE can therefore be used in a SQL statement, and be replaced by the current date when the value is accessed by commands such as Select or Insert. We will use a view of the Orders table called Entryorders to insert rows into the Orders table. Create the Orders table, so that SYSDATE is automatically inserted for odate if no value is given. Then create the Entryorders view so that no values can be given.

```
CREATE TABLE Orders1 ( Onum INT PRIMARY KEY,   Amt
DECIMAL(10,2), Odate TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
Cnum INT,Snum INT );

CREATE VIEW Entryorders AS

SELECT Onum, Amt, Cnum, Snum

FROM Orders;


INSERT INTO Entryorders (Onum, Amt, Cnum, Snum)

VALUES (3012, 2500.00, 2005, 1003);

Select * from Entryorders;
```

Assignment - 21 (Grant and Revoke)

1) Give Amit the right to change the ratings of the customers.

```
GRANT UPDATE (rating) ON Customers TO 'Amit'@'your_host';
```

2) Give Manoj the right to give other users right to query the Orders table.

```
GRANT SELECT ON Orders TO 'Manoj'@'localhost';
```

3) Take the INSERT privilege on Salespeople away from Ajita.

```
REVOKE INSERT ON Salespeople FROM 'Ajita'@'localhost';
```

4) Grant Abhijeet the right to insert or update the Customers table while keeping her possible rating values in the range of 100 to 500.

```
GRANT INSERT, UPDATE ON Customers TO 'Abhijeet'@'localhost';
```

5) Allow Vikram to query the Customers table, but restrict his access to those customers whose rating is the lowest.

```
CREATE VIEW LowestRatedCustomers AS

SELECT * FROM Customers

WHERE rating = (SELECT MIN(rating) FROM Customers);

GRANT SELECT ON LowestRatedCustomers TO 'Vikram'@'localhost';
```