**LAB CYCLE 2**

**QUESTION SET 1**

Create the following tables and execute the queries given below

**SAILORS**

|  |  |  |  |
| --- | --- | --- | --- |
| **sid** | **sname** | **rating** | **age** |
| 22 | Dustin | 7 | 45 |
| 29 | Brutas | 1 | 33 |
| 31 | Lubber | 8 | 55 |
| 32 | Andy | 8 | 25 |
| 58 | Rusty | 10 | 35 |
| 64 | Horatio | 7 | 35 |
| 71 | Zorba | 10 | 16 |
| 74 | Horatio | 9 | 35 |
| 85 | Art | 3 | 26 |
| 95 | Bob | 3 | 64 |

**BOATS**

|  |  |  |
| --- | --- | --- |
| **Bid** | **bname** | **color** |
| 101 | Interlake | Blue |
| 102 | Interlake | Red |
| 103 | Clipper | Green |
| 104 | Marine | Red |

**RESERVES**

|  |  |  |
| --- | --- | --- |
| **sid** | **bid** | **day** |
| 22 | 101 | 10/10/98 |
| 22 | 102 | 10/10/98 |
| 22 | 103 | 10/8/98 |
| 22 | 104 | 10/7/98 |
| 31 | 102 | 11/10/98 |
| 31 | 103 | 11/6/98 |
| 31 | 104 | 11/12/98 |
| 64 | 101 | 9/5/98 |
| 64 | 102 | 9/8/98 |
| 74 | 103 | 9/8/98 |

**Query:** CREATE TABLE SAILORS

(sid int,sname VARCHAR(10),rating int,age int);

INSERT INTO SAILORS VALUES(22, 'Dustin', 7, 45),(29, 'Brutas', 1, 33),(31,

'Lubber', 8, 55),(32, 'Andy', 8, 25),(58, 'Rusty', 10, 35),(64, 'Horatio', 7, 35),(71,

'Zorba', 10, 16),(74, 'Horatio', 9, 35),(85, 'Art', 3, 26),(95, 'Bob', 3, 64);

SELECT \* FROM SAILORS;

CREATE TABLE BOATS

(

Bid NUMERIC(3),

bname VARCHAR(10),

color VARCHAR(7)

);

INSERT INTO BOATS VALUES(101, 'Interlake', 'Blue'),(102, 'Interlake',

'Red'),(103, 'Clipper', 'Green'),(104, 'Marine', 'Red');

SELECT \* FROM BOATS;

CREATE TABLE RESERVES

(

sid NUMERIC(2),

bid NUMERIC(3),

day DATE

);

INSERT INTO RESERVES VALUES

(22, 101, '98-10-10'),

(22, 102, '98-10-10'),

(22, 103, '98-08-10'),

(22, 104, '98-07-10'),

(31, 102, '98-10-11'),

(31, 103, '98-06-11'),

(31, 104, '98-12-11'),

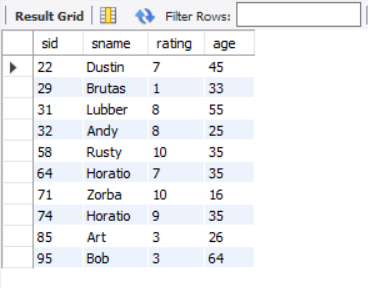
(64, 101, '98-05-09'),

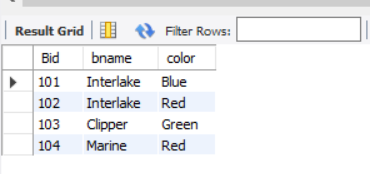
(64, 102, '98-08-09'),

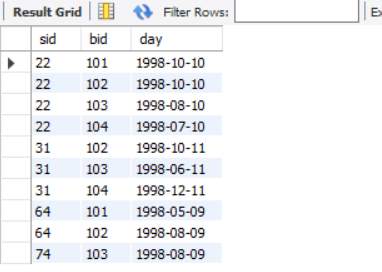
(74, 103, '98-08-09');

SELECT \* from RESERVES;

**Output:**



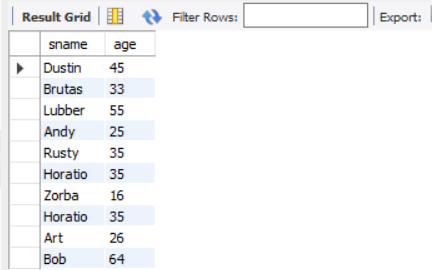




1.Find the names and ages of all sailors

**Query:** SELECT sname, age FROM sailors;

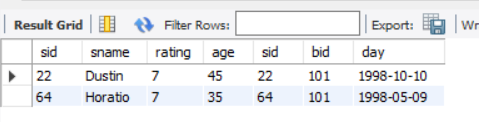
**Output:**



2. Find all information of sailors who have reserved boat number 101.

**Query:** SELECT \* FROM SAILORS,RESERVES WHERE sailors.sid = reserves.sid AND bid = 101;

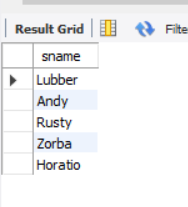
**Output:**



3. Find all sailors with rating above 7

**Query:** SELECT sname FROM SAILORS WHERE rating > 7;

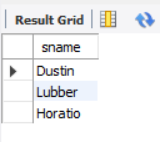
**Output:**



4. Find the names of sailors who have reserved boat no 103

**Query:** SELECT sname FROM SAILORS,RESERVES WHERE bid = 103 AND sailors.sid = reserves.sid;

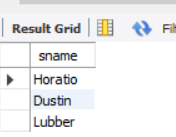
**Output:**



5. Find the names of sailors who have reserved a red boat, and list in the order of age.

**Query:** SELECT DISTINCT sname FROM SAILORS,RESERVES,BOATS WHERE color = 'Red' AND boats.bid = reserves.bid AND sailors.sid = reserves.sid order by age;

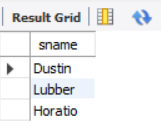
**Output:**



6. Find the names of sailors who have reserved either a red or green boat

**Query:** SELECT DISTINCT sname FROM SAILORS,RESERVES,BOATS WHERE ( color = 'Red'OR color = 'Green') AND boats.bid = reserves.bid AND sailors.sid = reserves.sid;

**Output:**

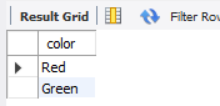


7. Find the colors of boats reserved by “Lubber”.

**Query:** SELECT DISTINCT color FROM SAILORS,RESERVES,BOATS

WHERE sname = 'Lubber'AND boats.bid = reserves.bid AND sailors.sid = reserves.sid;

**Output:**

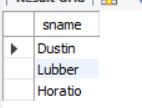


8. Find the names of sailors who have reserved both red and green boats

**Query:** SELECT DISTINCT sname FROM SAILORS,RESERVES,BOATS WHERE color = 'Red' AND boats.bid = reserves.bid AND sailors.sid = reserves.sid UNION

SELECT DISTINCT sname FROM SAILORS,RESERVES,BOATS WHERE color = 'Green' AND boats.bid = reserves.bid AND sailors.sid = reserves.sid;

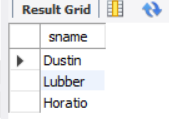
**Output:**



9. Find the names of sailors who have reserved at least one boat

**Query:** SELECT DISTINCT sname FROM SAILORS,RESERVES WHERE reserves.sid = sailors.sid;

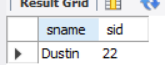
**Output:**



10. Find the ids and names of sailors who have reserved two different boats on the same day.

**Query:** SELECT sname,reserves.sid FROM RESERVES,SAILORS WHERE sailors.sid = reserves.sid GROUP BY DAY,reserves.sid,sname HAVING COUNT(DAY)>1 ;

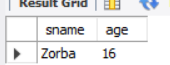
**Output:**



11. Find the name and the age of the youngest sailor.

**Query:** SELECT sname,age FROM SAILORS WHERE age =(SELECT MIN(age) FROM sailors) ;

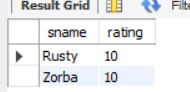
**Output:**



12. Find the names and ratings of sailor whose rating is better than some sailor called Horatio.

**Query:** SELECT sname,rating FROM SAILORS WHERE rating >(SELECT MAX(rating) FROM SAILORS WHERE sname = 'Horatio');

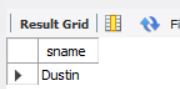
**Output:**



13. Find the names of sailors who have reserved all boats.

**Query:** SELECT sname FROM(SELECT sname,reserves.sid,COUNT(bid) AS id FROM RESERVES,SAILORS WHERE reserves.sid = sailors.sid GROUP BY reserves.SID,sname) a WHERE id =( SELECT COUNT(bid) FROM BOATS);

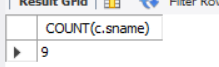
**Output:**



14. Count the number of different sailor names.

**Query:** SELECT COUNT(c.sname)FROM(SELECT DISTINCT sname FROM Sailors) c;

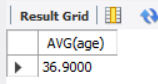
**Output:**



15. Calculate the average age of all sailors.

**Query:** SELECT AVG(age) FROM SAILORS;

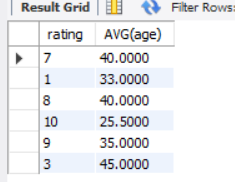
**Output:**



16. Find the average age of sailors for each rating level.

**Query:** SELECT rating,AVG(age) FROM SAILORS GROUP BY rating;

**Output:**

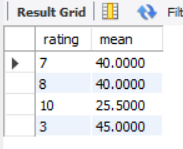


17. Find the average age of sailors for each rating level that has at least two sailors.

**Query:** SELECT a.rating,b.mean FROM(SELECT COUNT(sname) AS num,rating FROM SAILORS GROUP BY rating HAVING COUNT(sname)>1)a,

(SELECT rating,AVG(age) AS mean FROM SAILORS GROUP BY rating ) b WHERE a.rating = b.rating;

**Output:**



**QUESTION SET 2**

1. Create the table STUDENT\_INFO with Columns: Sid, Stud\_name & stude\_score.

* Insert values into STUDENT\_INFO with the following constraints:Sid should be unique, Stud name NOT NULL and stude\_score DEFAULT value of 20.
* Set Sid as primary key.
* Update stude\_score by adding a value of 5 to stude\_score in the table STUDENT\_INFO for the rows satisfying the condition of stude\_score >150 (Using CASE)

**Query:** CREATE TABLE STUDENT\_INFO(Sid INT UNIQUE ,Stud\_name VARCHAR(20) NOT NULL,stude\_score numeric(5,2) DEFAULT 20);

ALTER TABLE STUDENT\_INFO ADD PRIMARY KEY (Sid);

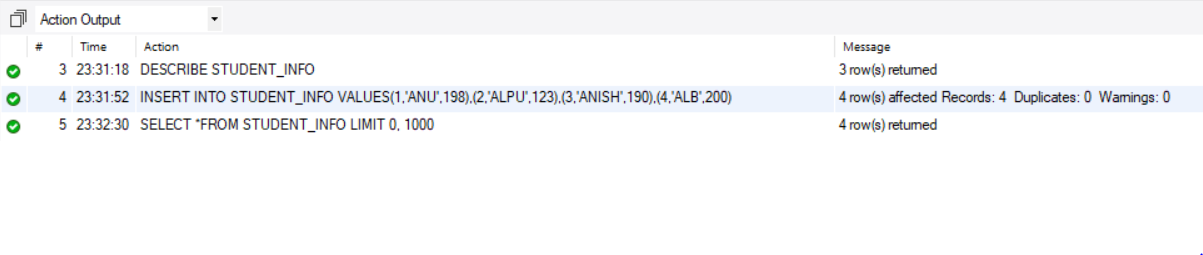
DESCRIBE STUDENT\_INFO;

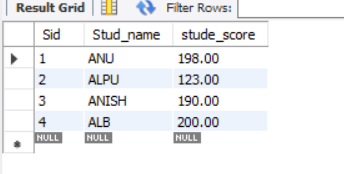
INSERT INTO STUDENT\_INFO VALUES(1,'ANU',198),(2,'ALPU',123),(3,'ANISH',190),(4,'ALB',200);

UPDATE STUDENT\_INFO SET Stude\_score = Stude\_score + 5 WHERE Stude\_score > 150;

SELECT \*FROM STUDENT\_INFO;

**Output:**





2. Create the tables **worker** and **bonus** with the following fields. The primary key of Worker table is Worker\_ID. Set Worker\_id as foreign key of bonus on update and delete cascade constraints. Each constraint should be given a name

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **WORKER\_ID** | **FIRST\_NAME** | **LAST\_NAME** | **SALARY** | **JOINING\_DATE** | **DEPARTMENT** |
| 1 | Monika | Arora | 100000 | 2014-02-20 | HR |
| 2 | Niharika | Verma | 80000 | 2014-06-11 | Admin |
| 3 | Vishal | Singhal | 300000 | 2014-02-20 | HR |
| 4 | Amitabh | Singh | 500000 | 2014-02-20 | Admin |
| 5 | Vivek | Bhati | 500000 | 2014-06-11 | Admin |
| 6 | Vipul | Diwan | 200000 | 2014-06-11 | Account |
| 7 | Satish | Kumar | 75000 | 2014-01-20 | Account |
| 8 | Geetika | Chauhan | 90000 | 2014-04-11 | Admin |

**Query:** CREATE TABLE worker

( Worker\_ID INT,

first\_name VARCHAR(15),

last\_name VARCHAR(15),

salary NUMERIC(8),

joining\_date DATE,

department VARCHAR (15) );

ALTER TABLE worker ADD PRIMARY KEY (Worker\_ID);

**Output:**



**Query:** INSERT INTO worker VALUES (1,'monika','arora',100000,'2014-02-20','hr'),

(2,'niharika','verma',80000,'2014-06-11','admin'),

(3,'vishal','singhal',300000,'2014-02-20','hr'),

(4,'amithabh','singh',500000,'2014-02-20','admin'),

(5,'vivek','bhati',500000,'2014-06-11','admin'),

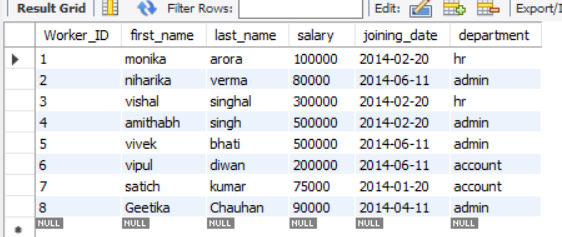
(6,'vipul','diwan',200000,'2014-06-11','account'),

(7,'satich','kumar',75000,'2014-01-20','account'),

(8,'Geetika','Chauhan',90000,'2014-04-11','admin');

select \*from worker;

**Output:**



3. **Sample Table – Bonus**

|  |  |  |
| --- | --- | --- |
| **WORKER\_ID** | **BONUS\_DATE** | **BONUS\_AMOUNT** |
| 1 | 2016-02-20 | 5000 |
| 2 | 2016-06-11 | 3000 |
| 3 | 2016-02-20 | 4000 |
| 1 | 2016-02-20 | 4500 |
| 2 | 2016-06-11 | 3500 |

**Query:** CREATE TABLE bonus

( Worker\_ID INT, bonus\_date DATE,

bonus\_amount NUMERIC(6) );

ALTER TABLE bonus ADD CONSTRAINT fk\_cod\_csd

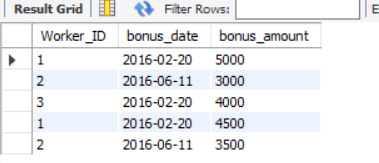
FOREIGN KEY(Worker\_ID) REFERENCES

worker(Worker\_ID) ON DELETE CASCADE ;

INSERT INTO bonus VALUES (1,'2016-02-20',5000),(2,'2016-06-11',3000),(3,'2016-02-20',4000),(1,'2016-02-20',4500),(2,'2016-06-11',3500);

select \*from bonus;

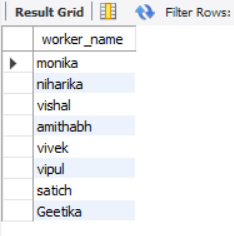
**Output:**



4. Write An SQL Query To Fetch “FIRST\_NAME” From Worker Table Using The Alias Name As <WORKER\_NAME>.

**Query:** SELECT first\_name AS worker\_name FROM worker;

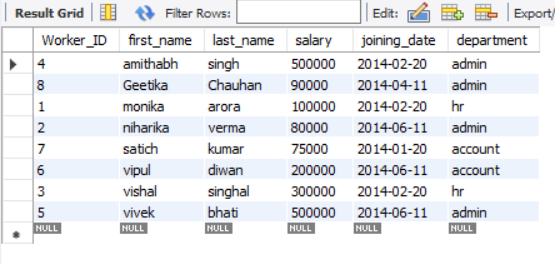
**Output:**



5. Write An SQL Query To Print All Worker Details From The Worker Table Order By FIRST\_NAME Ascending

**Query:** SELECT \* FROM worker ORDER BY trim(first\_name) ASC ;

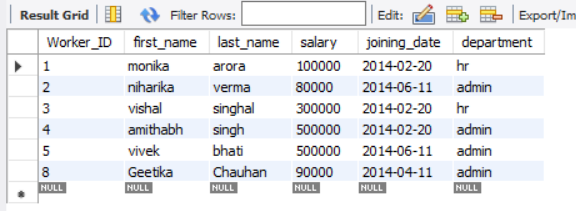
**Output:**



6. Write An SQL Query To Print Details Of Workers Excluding First Names, “Vipul” And “Satish” From Worker Table.

**Query:** SELECT \* FROM worker WHERE trim(first\_name) != 'vipul' AND trim(first\_name) != 'satich';

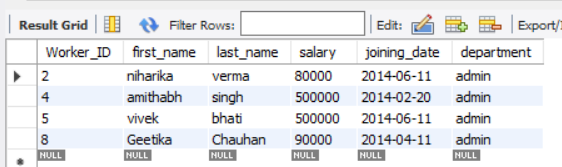
**Output:**



7. Write An SQL Query To Print Details Of Workers With DEPARTMENT Name As “Admin”.

**Query:** select \*from worker where department='admin';

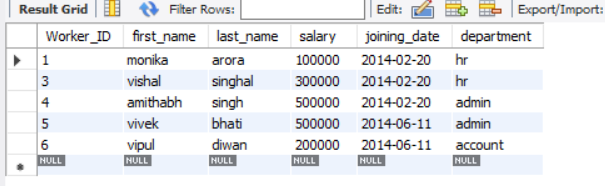
**Output:**



8. Write An SQL Query To Print Details Of The Workers Whose SALARY Lies Between 100000 And 500000

**Query:** SELECT \* FROM worker WHERE salary BETWEEN 100000 AND 500000;

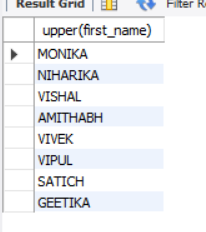
**Output:**



9. Write An SQL Query To Fetch “FIRST\_NAME” From Worker Table In Upper Case. (upper())

**Query:** SELECT upper(first\_name) FROM worker;

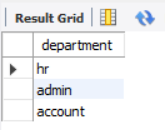
**Output:**



10. Write An SQL Query To Fetch Unique Values Of DEPARTMENT From Worker Table.

**Query:** SELECT distinct department FROM worker;

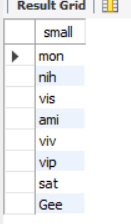
**Output:**



11. Write An SQL Query To Print First Three Characters Of  FIRST\_NAME From Worker Table.( substring())

**Query:** SELECT SUBSTR(first\_name, 1, 3) AS small FROM worker;

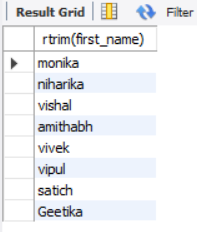
**Output:**



12. Write An SQL Query To Print The FIRST\_NAME From Worker Table After Removing White Spaces From The Right Side( RTRIM ( ))

**Query:** SELECT rtrim(first\_name) FROM worker;

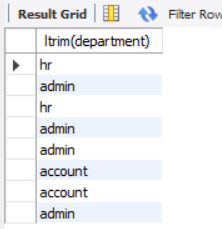
**Output:**



13. Write An SQL Query To Print The DEPARTMENT From Worker Table After Removing White Spaces From The Left Side. ( LTRIM ( ))

**Query:** SELECT ltrim(department) FROM worker;

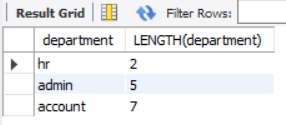
**Output:**



14. Write An SQL Query That Fetches The Unique Values Of DEPARTMENT From Worker Table And Prints Its Length.( length())

**Query:** SELECT distinct department, LENGTH(department) FROM worker ;

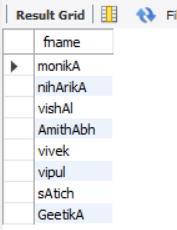
**Output:**



15. Write An SQL Query To Print The FIRST\_NAME From Worker Table After Replacing ‘a’ With ‘A’.( REPLACE( ))

**Query:** SELECT REPLACE(first\_name, 'a', 'A') AS fname FROM worker;

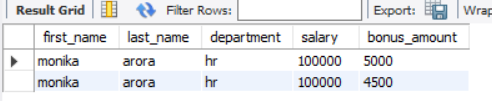
**Output:**



16. Find the First name , last name ,Department, Salary and Bonus of employees whose bonus amount is greater than 4000

**Query:** SELECT first\_name,last\_name,department,salary,bonus\_amount FROM worker,bonus WHERE worker.worker\_id = bonus.worker\_id AND bonus\_amount > 4000 ;

**Output:**

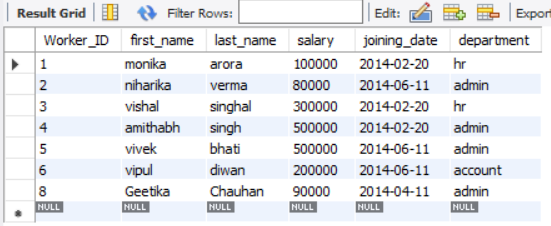


17. Delete the employee with worker\_id=7 from worker and display the details of both tables.

**Query:** DELETE from worker WHERE worker\_id=7;

SELECT \* FROM worker;

**Output:**



18. Drop the foreign key constraint and add a new referential integrity constraint with ‘on update or delete with no action’

**Query:** ALTER TABLE bonus DROP CONSTRAINT fk\_cod\_csd;

ALTER TABLE bonus ADD CONSTRAINT fk\_cod\_na FOREIGN KEY(Worker\_ID)

REFERENCES worker(Worker\_ID) ON DELETE no action ;

**Output:**

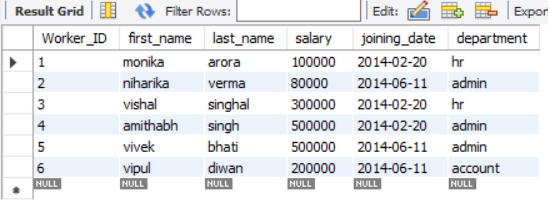


19. Delete the employee with worker\_id = 8 from worker.

**Query:** DELETE from worker WHERE worker\_id=8;

SELECT \* FROM worker;

**Output:**



**QUESTION SET 3**

Create the tables given below and execute the queries:

**Customer(Cust id : integer, cust\_name: string)**

**Item(item\_id: integer, item\_name: string, price: integer)**

**Sale(bill\_no: integer, bill\_date: date, cust\_id: integer, item\_id: integer, qty\_sold: integer)**

For the above schema, perform the following—

a) Create the tables with the appropriate integrity constraints

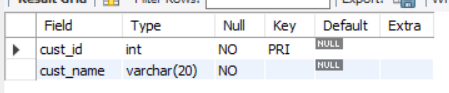
**Query:** CREATE TABLE customer

(cust\_id INT PRIMARY KEY,

cust\_name VARCHAR(20) NOT NULL);

describe customer;

**Output:**



**Query:** CREATE TABLE item

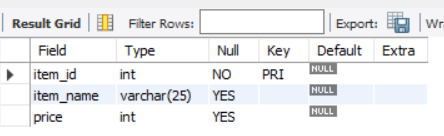
(item\_id INT PRIMARY KEY,

item\_name VARCHAR(25),

price INT);

describe item;

**Output:**



**Query:** CREATE TABLE sale

(bill\_no INT PRIMARY KEY,

bill\_date DATE NOT NULL,

cust\_id INT NOT NULL,

item\_id INT NOT NULL,

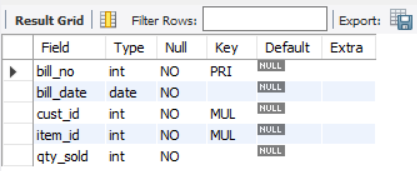
qty\_sold INT NOT NULL,

FOREIGN KEY(item\_id) REFERENCES item(item\_id),

FOREIGN KEY (cust\_id) REFERENCES customer(cust\_id));

describe sale;

**Output:**



b) Insert details of 5 customers, 5 items and 10 sales details. There should be one customer ‘rekha’ who had purchased 3 different products on the same date. And there should be atleast one customer who had purchased 2 different products on the same date in the year ‘2018’.

**Query:** INSERT INTO customer VALUES

(1,'Rekha'),

(2,'Rani'),

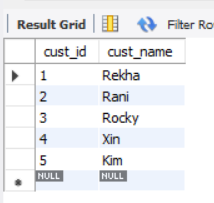
(3,'Rocky'),

(4,'Xin'),

(5,'Kim');

select \* from customer;

**Output:**



**Query:** INSERT INTO item VALUES

(1,'Rusk',120),

(2,'Banana',50),

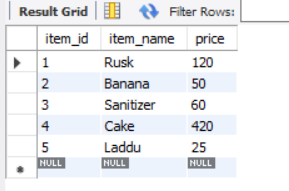
(3,'Sanitizer',60),

(4,'Cake',420),

(5,'Laddu',25 );

select \* from item;

**Output:**



**Query:** INSERT INTO sale VALUES

(10, '2020-10-01', 1, 1, 3),

(11, '2020-10-01', 1, 3, 2),

(12, '2020-10-01', 1, 5, 7),

(13, '2018-10-01', 4, 4, 1),

(14, '2018-10-11', 4, 2, 2),

(15, '2018-09-29', 5, 1, 5),

(16, '2019-12-25', 3, 1, 5),

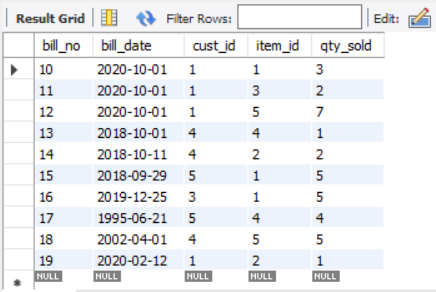
(17, '1995-06-21', 5, 4, 4),

(18, '2002-04-01', 4, 5, 5),

(19, '2020-02-12', 1, 2, 1);

SELECT \* FROM sale;

**Output:**



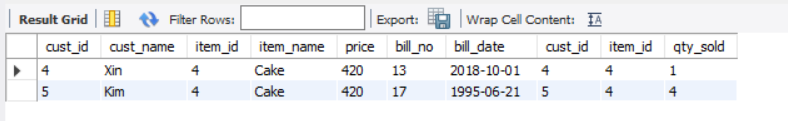
c) List the details of the customer who have bought a product which has a price>200

**Query:** SELECT \* FROM customer,item,sale WHERE price > 200

AND sale.item\_id = item.item\_id

AND sale.cust\_id = customer.cust\_id;

**Output:**



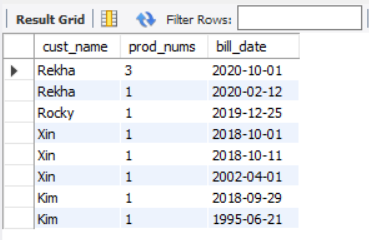
d) Give a count of how many products have been bought by each customer group by bill date.

**Query:** SELECT cust\_name,prod\_nums,bill\_date FROM customer,

(SELECT cust\_id,COUNT(item\_id) AS prod\_nums,bill\_date FROM sale

GROUP BY bill\_date) ci WHERE ci.cust\_id = customer.cust\_id;

**Output:**



e) Give a count of how many products have been bought by each customer group by bill date only for the year 2018.

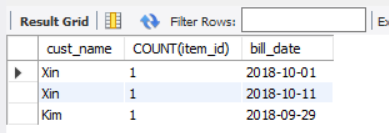
**Query:** SELECT cust\_name,COUNT(item\_id),bill\_date FROM customer c,sale s

WHERE c.cust\_id = s.cust\_id

GROUP BY cust\_name,bill\_date

HAVING extract(YEAR FROM bill\_date) = 2018;

**Output:**



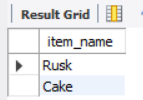
f) Give a list of products bought by a customer having cust\_id as 5

**Query:** SELECT item\_name FROM item,sale

WHERE sale.item\_id = item.item\_id

AND sale.cust\_id = 5;

**Output:**



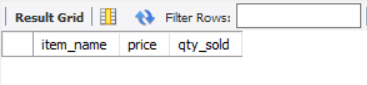
g) List the item details which are sold as of today

**Query:** SELECT item\_name,price,qty\_sold FROM item ,sale

WHERE sale.item\_id = item.item\_id

AND bill\_date = CURDATE();

**Output:**



h) Print the bill in a neat format with the quantity sold, price of the item and the final amount of customer ‘rekha’

**Query:** SELECT item\_name,qty\_sold,price,(qty\_sold \* price) AS total\_amount

FROM customer,item,sale

WHERE cust\_name = 'Rekha'

AND sale.item\_id = item.item\_id

AND sale.cust\_id = customer.cust\_id;

**Output:**

