Name	CWID

Homework Assignment 2

Due Date: October 12, 2019 (11:59PM CDT)

CS425 - Database Organization Results

Please leave this empty!

2.1		2.2													
2.3		2.4		2.5		2.6		2.7		2.8		2.9		2.10	
2.11		2.12		2.13		2.14								Sum	

Instructions

- Try to answer all the questions using what you have learned in class
- **When writing a query, write the query in a way that it would work over all possible database instances and not just for the given example instance!**
- After creating the relations, you can make up some data and insert them into the relations
- Please submit the homework (as a single sql file **your_name.sql**) electronically using blackboard
- In the **your_name.sql** file, make sure you have marked the answer for each question using comments
- All the sql scripts should be **executable** in Oracle 11g (e.g., on fourier) or recent versions

Consider the following database schema and example instance:

Employee

<u>eid</u>	ename	dob	salary
0011	Alice	07/03/1976	140,000
0012	Bob	09/14/1978	110,000
0013	Chris	06/22/1991	80,000
0014	David	05/17/1982	105,000
0015	Edward	04/15/1988	98,000
...	...	...	...

Customer

<u>cid</u>	cname	city	state
B101	Citi Bank	New York	NY
B102	JP Morgan	New York	NY
B103	Fidelity	Boston	MA
B104	Moto	Chicago	IL
B105	Disney	Orlando	FL
...	...	...	...

Product

<u>pid</u>	pname	unit_price
A17811	Desk	750
A17812	Chair	230
A23453	PC	900
A34451	Mac	300
...	...	...

Sales

<u>eid</u>	<u>cid</u>	<u>pid</u>	quantity
0011	B101	A17811	700
0011	B102	A17812	200
0012	B102	A23453	100
0013	B101	A34451	50
...	...	...	...

Manages

<u>eid1</u>	<u>eid2</u>
0011	0013
0013	0017
0012	0013
...	...

Hints:

- Underlined attribute(s) form the primary key of a relation.
- Relation *Sales* stores the sales information: the salesman (employee ID) who is in charge of the ordered product, the product ID, the customer ID, and the quantities of ordered products.
- The attributes *eid*, *cid* and *pid* of relation *Sales* are the foreign keys to relations *Employee* (salesman), *Customer* and *Product*, respectively.
- Relation *Manages* stores the "managing" information in the company (e.g., the employee with *eid1* is the direct manager of the employee with *eid2*).
- All the IDs' are strings.

Part 2.1 SQL DDL (Total: 20 Points)

Question 2.1.1 (15 Points)

Write the SQL statements for all the five relations: *Employee*, *Customer*, *Product*, *Sales*, and *Manages*. Note: (1) An appropriate data type should be used for each attribute, and (2) All the foreign keys should be created.

Solution

```sql
CREATE TABLE Employee(
eid VARCHAR(4),
ename VARCHAR(20),
dob DATE,
salary INT,
PRIMARY KEY(eid)
);

CREATE TABLE Customer(
cid VARCHAR(4),
cname VARCHAR(15),
city VARCHAR(15),
state CHAR(2),
PRIMARY KEY(cid)
);

CREATE TABLE Product(
pid VARCHAR(6),
pname VARCHAR(20),
unit_price INT,
PRIMARY KEY(pid)
);

CREATE TABLE Sales(
eid VARCHAR(4),
cid VARCHAR(4),
pid VARCHAR(6),
quantity INT,
PRIMARY KEY(eid,cid,pid),
FOREIGN KEY(eid) REFERENCES Employee,
FOREIGN KEY(cid) REFERENCES Customer,
FOREIGN KEY(pid) REFERENCES Product
);

CREATE TABLE Manages(
eid1  VARCHAR(4),
eid2  VARCHAR(4),
PRIMARY KEY (eid1, eid2)
);
```

Question 2.1.2 (5 Points)

Write an SQL statement that adds a constraint to the *Employee* relation to make sure that the *salary* attribute cannot be `NULL`, and the value of this attribute has to be between 40,000 and 500,000. Furthermore, the default value for this attribute should be 80,000.

Solution

```
ALTER TABLE Employee
MODIFY salary INT DEFAULT 80000 NOT NULL
ADD CONSTRAINT salary_check CHECK (salary >= 40000 AND salary <= 500000);
```

OR

```
ALTER TABLE Employee MODIFY salary INT NOT NULL;
ALTER TABLE Employee MODIFY salary INT DEFAULT 80000;
ALTER TABLE Employee ADD CONSTRAINT salary_check CHECK (salary >= 40000
AND salary <= 500000);
```

Part 2.2 SQL Queries (Total: 56 Points)

Question 2.2.1 (7 Points)

Write an SQL query that returns the products (pid and pname) which are sold by the direct manager(s) of employee ‘0013’ and the quantity for any customer (not aggregated) is greater than 300. Hint: relation *Manages* stores the “managing” information in the company (e.g., the employee with *eid1* is the direct manager of the employee with *eid2*).

Solution

```
SELECT pid, pname
FROM Product NATURAL JOIN Sales
WHERE quantity > 300 AND eid IN
(SELECT eid1 AS eid FROM Manages WHERE eid2 = '0013'
);
```

Question 2.2.2 (7 Points)

Write an SQL query that returns the IDs of all the products, with the total ordered quantity in ‘Chicago’ greater than the total ordered quantity in ‘Boston’.

Solution

```
WITH C_quantity AS
   (SELECT pid, sum(quantity) AS total_quantity
   FROM Sales NATURAL JOIN Customer
   WHERE city = 'Chicago'
   GROUP BY pid),
   B_quantity AS
   (SELECT pid, sum(quantity) AS total_quantity
   FROM Sales NATURAL JOIN Customer
   WHERE city = 'Boston'
   GROUP BY pid)
   SELECT C.pid FROM C_quantity C, B_quantity B
   WHERE C.pid = B.pid AND C.total_quantity > B.total_quantity;
```

Question 2.2.3 (7 Points)

Write an SQL query that returns the name and unit price of all the products (pname and unit_price) which are never ordered by ‘Disney’.

Solution

```
SELECT pname, unit_price
FROM Product WHERE pid
NOT IN
   (SELECT pid FROM Sales NATURAL JOIN Customer
      WHERE cname = 'Disney');
```

Question 2.2.4 (7 Points)

Write an SQL query that returns all the employees and their average sales amount (of all the customers) for each of the products (query result: eid, pid and avg_sales).

Solution

```
SELECT eid, pid, avg(quantity * unit_price) AS avg_sales
FROM Sales NATURAL JOIN Product NATURAL JOIN Employee
GROUP BY eid, pid;
```

Question 2.2.5 (7 Points)

Write an SQL query that returns all the employees and the total number of employees directly managed by each employee (if no one is directly managed, then return 0). Hint: relation *Manages* stores the “managing” information in the company (e.g., the employee with *eid1* is the direct manager of the employee with *eid2*).

Solution

```
WITH m AS
(SELECT * FROM Employee JOIN Manages ON Employee.eid = Manages.eid1),
notm AS
(SELECT * FROM Employee
WHERE NOT  EXISTS (SELECT eid1 AS eid FROM Manages)
)
SELECT eid, COUNT(*) FROM m GROUP BY eid
UNION
SELECT eid, 0 FROM notm;
```

Question 2.2.6 (7 Points)

Write an SQL query that returns the number of products where the average ordered quantity (of all the employees and customers) is lower than 1000.

Solution

```
WITH p_avg_quantity (pid, avg_quantity) AS
(SELECT pid, AVG(quantity) AS avg_quantity
FROM sales
GROUP BY pid
HAVING AVG(quantity) < 1000)
SELECT COUNT(*) FROM p_avg_quantity;
```

Question 2.2.7 (7 Points)

Write an SQL query that returns the product names and the name of the customer(s) who ordered the highest quantity (from all the employees). Query result: pname and cname.

Solution

```
WITH total_q(pid, cid, tq) AS
(SELECT pid, cid, SUM(quantity) FROM Sales
GROUP BY pid, cid),
high_q AS
(SELECT MAX(tq) AS hq
FROM total_q)
SELECT pname, cname
FROM (SELECT pid, cid, tq FROM total_q WHERE tq=(SELECT hq FROM high_q))
NATURAL JOIN Product
NATURAL JOIN Customer;
```

Question 2.2.8 (7 Points)

Write an SQL query which returns all the employees' IDs and their managed employees' IDs. Hint: the result should include not only directly managed employees but also indirectly managed employees.

Solution

```
WITH rec_manage(eid1, eid2) AS (
   SELECT eid1, eid2 FROM Manages
UNION ALL
   SELECT rec_manage.eid1, Manages.eid2
   FROM rec_manage, Manages WHERE rec_manage.eid2 = Manages.eid1
   )
SELECT *
FROM rec_manage;
```

Note: using left outer join for listing the employees without any managed employee is also considered as correct for this question.

Part 2.3 SQL Updates (Total: 24 Points)

Question 2.3.1 (5 Points)

Delete all the customers without placing any order.

Solution

```
DELETE FROM Customer
WHERE cid NOT IN (SELECT cid
                  FROM sales);
```

Question 2.3.2 (4 Points)

A new product *Telephone* is added to the warehouse (available for sale). The unit price is $100. Add the information to the *Product* relation. Assume that *pid* is automatically maintained by the system.

Solution

```
INSERT INTO Product(pname, unit_price)
VALUES('Telephone', 100);
```

Auto Increment for attributes can be realized by sequence/trigger in Oracle. See the examples:
https://chartio.com/resources/tutorials/how-to-define-an-auto-increment-primary-key-in-oracle/

Question 2.3.3 (8 Points)

Update the *unit_price* in the *Product* relation according to this rule:

- if it is negative, set it to 0
- if it is larger than 10,000, then set it to 2,500
- if it is NULL, set it to 1,000
- if none of the above applies do not change the *unit_price*

Note that we expect you to write a single statement that implements this.

Solution

```
UPDATE Product
SET unit_price =(CASE
                  WHEN unit_price < 0 THEN 0
                  WHEN unit_price > 10000 THEN 2500
                  WHEN unit_price IS NULL THEN 1000
                  END)
WHERE unit_price < 0 OR unit_price > 10000 OR unit_price IS NULL;
```

Question 2.3.4 (7 Points)

Update the salaries of employees as his/her current salary + 0.02 · (his/her total sales amount).

Solution

```
UPDATE Employee e
SET e.salary = e.salary + 0.02 *
(WITH total_amount(eid, amount) AS (SELECT Sales.eid, quantity*unit_price AS amount
FROM Sales NATURAL JOIN Product)
SELECT SUM(amount) AS bonus FROM total_amount WHERE total_amount.eid = e.eid)
WHERE e.eid IN (SELECT eid FROM Sales);
```