NETAJI SUBHAS UNIVERSITY OF TECHNOLOGY

NAME: SIDDHARTH GOENKA

ROLL NO: 2021UCM2371

BRANCH: MATHEMATICS AND COMPUTING

SECTION: MAC-1

EXPERIMENT 1

Consider the following relational schema

SAILORS (sid, sname, rating, date_of_birth)

BOATS (bid, bname, color)

RESERVES (sid, bid, date, time slot)

Write the following queries in SQL and relational algebra

1) Find sailors who've reserved at least one boat

2) Find names of sailors who've reserved a red or a green boat in the month of March.

3) Find names of sailors who've reserved a red and a green boat

4) Find sid of sailors who have not reserved a boat after Jan 2018.

5) Find sailors whose rating is greater than that of all the sailors named "John"

6) Find sailors who've reserved all boats

7) Find name and age of the oldest sailor(s)

8) Find the age of the youngest sailor for each rating with at least 2 such sailors

---

CREATION OF TABLES

```
-- creating table sailors
CREATE TABLE SAILORS(
SID INT PRIMARY KEY,
SNAME VARCHAR(20),
RATING INT,
DOB DATE
);

-- creating table boats

CREATE TABLE BOATS(
BID INT PRIMARY KEY,
```

```
BNAME VARCHAR(20),
COLOR VARCHAR(10)

);


-- creating table reserves

CREATE TABLE RESERVES(
SID INT NOT NULL,
BID INT NOT NULL,
DT DATE NOT NULL,
TIMESLOT INT,
FOREIGN KEY(SID)
        REFERENCES SAILORS(SID),
FOREIGN KEY(BID)
        REFERENCES BOATS(BID)

);
```

INSERTION OF VALUES

```
-- insert into sailors
INSERT INTO SAILORS(SID,SNAME,RATING,DOB) VALUES
(1,'ABC',7,'03-01-01'),
(2,'DEF',9,'12-07-01'),
(3,'GHI',9,'22-05-01'),
(4,'JKL',8,'23-01-01'),
(5,'MNO',8,'01-09-01');
```

BID	BNAME	COLOR
101	Interlake	blue
102	Interlake	red
103	Clipper	green
104	Marine	red
NULL	NULL	NULL

-- insert into boats

INSERT INTO BOATS(BID,BNAME,COLOR) VALUES

(101,'Interlake','blue'),

(102,'Interlake','red'),

(103,'Clipper','green'),

(104,'Marine','red');

SID	BID	DT	TIMESLOT
1	101	2010-10-21	1
1	102	2010-10-21	2
1	103	2010-10-21	2
1	104	2010-10-21	2
1	101	2010-10-21	1
2	102	2001-11-21	3
2	102	2007-11-21	3
3	101	2007-11-17	2
3	102	2007-10-17	2
4	103	2019-11-17	1
2	103	2019-11-17	3
2	101	2010-03-21	1
2	102	2007-03-21	2

-- insert into reserves

INSERT INTO RESERVES(SID,BID,DT,TIMESLOT) VALUES

(1,101,'10-10-21',1),

(1,102,'10-10-21',2),

(1,103,'10-10-21',2),

(1,104,'10-10-21',2),

(1,101,'10-10-21',1),

(2,102,'01-11-21',3),

(2,102,'07-11-21',3),

(3,101,'07-11-17',2),

(3,102,'07-10-17',2),

(4,103,'19-11-17',1),

(2,103,'19-11-17',3),

(2,101,'10-03-21',1), (2,102,'07-03-21',2);

QUERIES

-- Find sailors who have reserved at least one boat

SELECT SID, SNAME FROM SAILORS WHERE SID IN (SELECT SID FROM RESERVES);

SID	SNAME
1	ABC
2	DEF
3	GHI
4	JKL
NULL	NULL

-- Find names of sailors who've reserved a red or a green boat in the month of March.

SELECT SID,SNAME FROM SAILORS WHERE SID IN (SELECT R.SID FROM BOAT B, RESERVES R

WHERE R.BID=B.BID AND B.COLOR='RED' AND (SELECT EXTRACT(MONTH FROM R.DT)='03')

UNION

SELECT R2.SID FROM BOAT B2,RESERVES R2

WHERE R2.BID=B2.BID AND COLOR='GREEN' AND (SELECT EXTRACT(MONTH FROM R.DT)='03')

);

sname
ABC
DEF
GHI
JKL

-- Find names of sailors who've reserved a red and a green boat

SELECT DISTINCT S1.SNAME FROM SAILORS S1, RESERVES R1 ,BOATS B1,RESERVES R2,BOATS B2

WHERE S1.SID=R1.SID AND R1.SID=B1.BID

AND S1.SID=R2.SID AND R2.VID=B2.BID

AND B1.COLOR='RED' AND B2.COLOR='GREEN';

sname
ABC
DEF

-- Find sid of sailors who have not reserved a boat after Jan 2018.

SELECT SID FROM SAILORS WHERE SID NOT IN (SELECT SID FROM RESERVES WHERE DT='01-01-18');

SID
1
2
3
4
5
NULL

-- Find sailors whose rating is greater than that of all the sailors named “John”

SELECT SNAME FROM SAILORS

WHERE RATING>ALL(SELECT RATING FROM SAILORS WHERE SNAME='MNO');

SNAME
DEF
GHI

-- Find sailors who’ve reserved all boats

SELECT SNAME FROM SAILORS S WHERE NOT EXISTS

(SELECT * FROM BOAT B WHERE NOT EXISTS

(SELECT * FROM RESERVES R WHERE R.SID=S.SID AND R.BID=B.BID));

sname
ABC

-- Find name and age of the oldest sailor(s)

SELECT SNAME,DATE_FORMAT(FROM_DAYS(DATEDIFF(now(),DOB)),'%Y') AS AGE FROM SAILORS ORDER BY DOB LIMIT 1;

	SNAME	AGE
▶	MNO	0021

-- Find the age of the youngest sailor for each rating with at least 2 such sailors

SELECT RATING,MAX(DATE_FORMAT(FROM_DAYS(DATEDIFF(now(),DOB)),'%Y')) AS MINAGE FROM SAILORS GROUP BY RATING HAVING COUNT(RATING)>=2;

	RATING	MINAGE
▶	9	0010
	8	0021

EXPERIMENT 2

Consider the following relational schema:

CUSTOMER (cust_num, cust_lname ,cust_fname, cust_balance);

PRODUCT (prod_num, prod_name, price)

INVOICE (inv_num, prod_num, cust_num, inv_date ,unit_sold, inv_amount);

Write SQL queries and relational algebraic expression for the following

1) Find the names of the customer who have purchased no item. Set default value of Cust_balance as 0 for such customers.

2) Write the trigger to update the CUST_BALANCE in the CUSTOMER table when a new invoice record is entered for the customer.

3) Find the customers who have purchased more than three units of a product on a day.

4) Write a query to illustrate Left Outer, Right Outer and Full Outer Join.

5) Count number of products sold on each date.

6) As soon as customer balance becomes greater than Rs. 100,000, copy the customer_num in new table called ”GOLD_CUSTOMER”

7) Add a new attribute CUST_DOB in customer table

CREATION OF TABLES

-- creating table customer

```
CREATE TABLE CUSTOMER(
CUST_NUM INT,
CUST_LNAME VARCHAR(50),
CUST_FNAME VARCHAR(50) NOT NULL,
CUST_BALANCE INT DEFAULT 0,
PRIMARY KEY(CUST_NUM)
);
```

-- creating table product

```
CREATE TABLE PRODUCT(
 PROD_NUM INT,
 PROD_NAME VARCHAR(50),
 PRICE INT NOT NULL,
 PRIMARY KEY(PROD_NUM)
);
```

-- creating table invoice

```
CREATE TABLE INVOICE(
INV_NUM INT,
PROD_NUM INT NOT NULL,
CUST_NUM INT NOT NULL,
INV_DATE DATE NOT NULL,
UNIT_SOLD INT NOT NULL,
INV_AMOUNT INT NOT NULL,
PRIMARY KEY(INV_NUM),
FOREIGN KEY(PROD_NUM) REFERENCES PRODUCT(PROD_NUM), FOREIGN KEY(CUST_NUM)
REFERENCES CUSTOMER(CUST_NUM),
CHECK(UNIT_SOLD>0));
```

INSERTION OF VALUES

-- inserting into customer

```
INSERT INTO CUSTOMER VALUES
(1,'SHARMA','AAYUSH',250),
(2,'KAPOOR','AKSHIT',1000),
(3,'VERMA','ARUN',0),
(4,'SHARMA','AVINASH',0),

(5,'RAJPUT','CHETAN',500),
(6,'DUGGAL','CHIRAG',800),
(7,'SINGH','MANDEEP',1000),
(8,'ANAND','NIKHIL',700);
```

-- INSERTING INTO PRODUCT

INSERT INTO PRODUCT VALUES

(1,'IPAD',60000),

(2,'EARPHONES',1800),

(3,'MOBILE PHONE', 20000),

(4,'LAPTOP',70000);

-- INSERTING INTO INVOICE

INSERT INTO INVOICE VALUES

(1,1,1,'2021-10-01',4,60000),

(2,1,2,'2021-10-02',2,2500),

(3,4,3,'2021-10-01',1,62000),

(4,3,4,'2021-10-03',3,22000),

(5,1,5,'2021-10-05',1,55000);

cust_num	cust_lname	cust_fname	cust_balance
1	sharma	aayush	250
2	kapoor	akshit	1000
3	verma	arun	0
4	sharma	avinash	0
5	rajput	chetan	500
6	duggal	chirag	800
7	singh	mandeep	1000
8	anand	nikhil	700
NULL	NULL	NULL	NULL

inv_num	prod_num	cust_num	inv_date	unit_sold	inv_amount
1	1	1	2021-10-01	4	60000
2	1	2	2021-10-02	2	2500
3	4	3	2021-10-01	1	62000
4	3	4	2021-10-03	3	22000
5	1	5	2021-10-05	1	55000
NULL	NULL	NULL	NULL	NULL	NULL

QUERIES

-- Names of the customer who have purchased no item

SELECT CONCAT(CUST_FNAME," ",CUST_LNAME) AS NAME FROM CUSTOMER
WHERE CUST_BALANCE = 0;

name
arun verma
avinash sharma

--Names of the customer who have purchased no item

SELECT CUST_NUM, CONCAT(CUST_FNAME," ",CUST_LNAME) AS NAME FROM CUSTOMER WHERE CUST_NUM IN(SELECT CUST_NUM FROM INVOICE GROUP BY CUST_NUM,INV_DATE,PROD_NUM

HAVING SUM(UNIT_SOLD)>3);

cust_num	name
1	aayush sharma

--Query to illustrate Left Outer, Right Outer and Full Outer Join.

SELECT CONCAT(C.CUST_FNAME,C.CUST_LNAME) AS NAME, I.INV_AMOUNT FROM CUSTOMER AS C LEFT JOIN INVOICE I ON C.CUST_NUM = I.CUST_NUM;

name	inv_amount
aayushsharma	60000
akshitkapoor	2500
arunverma	62000
avinashsharma	22000
chetanrajput	55000
chiragduggal	NULL
mandeepsingh	NULL
nikhilanand	NULL

--Number of products sold on each date.

SELECT INV_DATE,SUM(UNIT_SOLD) AS TOTAL_DSALES FROM INVOICE GROUP BY INV_DATE;

	inv_date	total_dsales
▶	2021-10-01	5
	2021-10-02	2
	2021-10-03	3
	2021-10-05	1

--As soon as customer balance becomes greater than Rs. 100,000, copy the customer_num in new table called ”GOLD_CUSTOMER”

```
CREATE TABLE GOLD_MASTER( CUST_NUM
INT,

        CUST_LNAME VARCHAR(50),
        CUST_FNAME VARCHAR(50),
        PRIMARY
        KEY(CUST_NUM) );

        DESC GOLD_MASTER; CREATE
        TRIGGER          IN_GOLD
           AFTER UPDATE ON CUSTOMER
        FOR
        EACH ROW
        INSERT INTO GOLD_MASTER

       (SELECT CUST_NUM,CUST_LNAME,CUST_FNAME

        FROM CUSTOMER
        WHERE CUST_NUM=NEW.CUST_NUM
        AND CUST_BALANCE>100000
        AND CUST_NUM NOT IN (SELECT CUST_NUM FROM GOLD_MASTER));
```

	Field	Type	Null	Key	Default	Extra
▶	cust_num	int	NO	PRI	NULL	
	cust_lname	varchar(50)	YES		NULL	
	cust_fname	varchar(50)	YES		NULL	

YES

--Add a new attribute CUST_DOB in customer table

ALTER TABLE CUSTOMER ADD COLUMN CUST_DOB DATE; DESC CUSTOMER;

Field	Type	Null	Key	Default	Extra
cust_num	int	NO	PRI	NULL	
cust_lname	varchar(20)	YES		NULL	
cust_fname	varchar(20 varchar(20)			NULL	
cust_balance	int	YES		0	
cust_dob	date	YES		NULL	

EXPERIMENT-3

Q 3: Consider the following relational schema

DEPARTMENT(Department_ID, Name, Location_ID) JOB (Job_ID ,

Function)

EMPLOYEE (Employee_ID, name, DOB, Job_ID , Manager_ID, Hire_Date, Salary, department_id) Answer the following queries using SQL and relational algebra:

a) Write a query to count number of employees who joined in March 2015

b) Display the Nth highest salary drawing employee details.

c) Find the budget (total salary) of each department.

d) Find the department with maximum budget.

e) Create a view to show number of employees working in Delhi and update it automatically when the database is modified.

f) Write a trigger to ensure that no employee of age less than 25 can be inserted in the database.

Creating tables

```
CREATE TABLE DEPARTMENT (
 Department_ID int PRIMARY KEY, Name
 varchar(20) NOT NULL, Location_ID INT
);
CREATE TABLE JOB ( Job_ID int
 PRIMARY KEY, Functions
 varchar(100)
);
CREATE TABLE EMPLOYEES (
 Employee_ID int,
 Name varchar(255),
 dob date,
 Job_ID int, Manager_ID
 int, Hire_Date date,
 Salary int, department_ID
 INT,
 FOREIGN KEY (Job_ID) REFERENCES JOB(Job_ID),
 FOREIGN KEY (Department_ID) REFERENCES department(Department_ID));
```

INSERTING VALUES

```
INSERT INTO DEPARTMENT VALUES (1,'Finance',302);
```

INSERT INTO DEPARTMENT VALUES (2,'Security',706);

INSERT INTO DEPARTMENT VALUES (3,'Human Resources',890);

INSERT INTO DEPARTMENT VALUES (4,'IT',6509);

INSERT INTO DEPARTMENT VALUES (5,'Electronic Dept',651);

INSERT INTO DEPARTMENT VALUES (6,'Software Dept',471);

INSERT INTO DEPARTMENT VALUES (7,'Hardware Dept',491);

←T→				Department_ID	Name	Location_ID
☐	Edit	Copy	Delete	1	0	302
☐	Edit	Copy	Delete	2	0	706
☐	Edit	Copy	Delete	3	0	890
☐	Edit	Copy	Delete	4	0	6509
☐	Edit	Copy	Delete	5	0	651
☐	Edit	Copy	Delete	6	0	471
☐	Edit	Copy	Delete	7	0	491

INSERT INTO JOB VALUES(501,'Manager of Department');

INSERT INTO JOB VALUES(502,'Works for Security team');

INSERT INTO JOB VALUES(503,'Works for Human Resources team');

INSERT INTO JOB VALUES(504,'works for IT team');

INSERT INTO JOB VALUES(505,'works for Finance team');

INSERT INTO JOB VALUES(506,'works for Electrical Dept team');

INSERT INTO JOB VALUES(507,'works for Hardware team');

INSERT INTO JOB VALUES(508,'works for Software team');

Employee_ID	Name	dob	Job_ID	Manager_ID	Hire_Date	Salary	department_ID
1001	Kartik	1999-03-14	502	801	0000-00-00	100000	2
1001	Himanshu	0000-00-00	502	801	2016-10-10	50000	2
1001	Rahul	1999-03-26	504	801	0000-00-00	60000	4
1001	Raj	1997-09-28	505	801	0000-00-00	50000	1
1001	Chetan	1996-11-17	505	801	0000-00-00	75000	1
1001	Sahil	1998-07-23	508	801	0000-00-00	87000	7
1001	Shiva	1997-12-05	502	801	0000-00-00	1000	2

INSERT INTO EMPLOYEES VALUES(1001,'Kartik','1999-03-14',502,801,'2015-0318',100000,2);

INSERT INTO EMPLOYEES VALUES(1001,'Himanshu','1997-05-16',502,801,'2016-10-10',50000,2);

INSERT INTO EMPLOYEES VALUES(1001,'Rahul','1999-03-26',504,801,'2015- 03-05',60000,4);

INSERT INTO EMPLOYEES VALUES(1001,'Raj','1997-09-28',505,801,'2018-10-11',50000,1);

INSERT INTO EMPLOYEES VALUES(1001,'Chetan','1996-11-17',505,801,'2014-0528',75000,1);

INSERT INTO EMPLOYEES VALUES(1001,'Sahil','1998-07-23',508,801,'2017- 12-15',87000,7);

INSERT INTO EMPLOYEES VALUES(1001,'Shiva','1997-12-05',502,801,'2019- 10-18',1000,2);

				Job_ID	Functions
☐	Edit	Copy	Delete	501	Manager of Department
☐	Edit	Copy	Delete	502	Works for Security team
☐	Edit	Copy	Delete	503	Works for Human Resources team
☐	Edit	Copy	Delete	504	works for IT team
☐	Edit	Copy	Delete	505	works for Finance team
☐	Edit	Copy	Delete	506	works for Electrical Dept team
☐	Edit	Copy	Delete	507	works for Hardware team
☐	Edit	Copy	Delete	508	works for Software team

QUERIES

a). SELECT COUNT(Employee_ID)

FROM Employee

WHERE hire_date> '2015-03-01' And hire_date<'2015-04-01';

	COUNT(Employee_ID)
▶	2

b). SELECT *

FROM employee

GROUP BY salary

ORDER BY salary DESC LIMIT 1;

	Employee_ID	Name	dob	Job_ID	Manager_ID	Hire_Date	Salary	department_ID
▶	1001	Kartik	1999-03-14	502	801	2015-03-18	100000	2

c) SELECT SUM(SALARY), DEPARTMENT_ID
FROM

EMPLOYEE

GROUP BY DEPARTMENT_ID;

	SUM(SALARY)	DEPARTMENT_ID
▶	125000	1
	151000	2
	60000	4
	87000	7

d). SELECT SUM(SALARY), DEPARTMENT_ID FROM

EMPLOYEE

GROUP BY DEPARTMENT_ID

Order by sum(salary) desc limit 1;

	SUM(SALARY)	DEPARTMENT_ID
▶	151000	2

e). CREATE VIEW DELHI_POPULATION AS SELECT
COUNT(EMPLOYEE_ID)FROM

EMPLOYEE,DEPARTMENT WHERE LOCATION_id=10;

	COUNT(EMPLOYEE_ID)
▶	0

f). delimiter $$

CREATE TRIGGER Check_age BEFORE INSERT ON employee FOR EACH

ROW

BEGIN

IF NEW.dob> 1993-01-01 THEN

SIGNAL SQLSTATE '45000' SET

MESSAGE_TEXT = 'ERROR:

AGE MUST BE ATLEAST 25 YEARS!';

```
END IF;
END;
```