# 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
١	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),

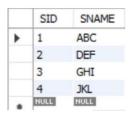
(3,101,'07-11-17',2),

```
(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);



-- 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')

);



-- 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';



-- 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
١	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');



-- 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));



-- 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
١	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),

-- INSERTING INTO PRODUCT

(8,'ANAND','NIKHIL',700);

# 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_Iname	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

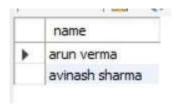
	inv_num	prod_num	cust_num	inv_date	unit_sold	inv_amount
•	1	1	1	2021-10-01	4	60000
	2	1	2	2021-10-02	2	2500
	3	4	3	2021-10-01	1	62000
	4	3	4	2021-10-03	3	22000
	5	1	5	2021-10-05	1	55000
	NULL	NULL	NULL	NULL	NULL	NULL

# **QUERIES**

-- Names of the customer who have purchased no item

SELECT CONCAT(CUST\_FNAME," ",CUST\_LNAME) AS NAME FROM CUSTOMER

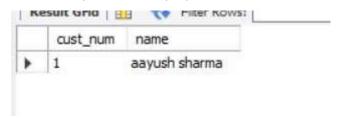
WHERE CUST\_BALANCE = 0;



--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);



--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;



--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
٠	2021-10-01	5
	2021-10-02	2
	2021-10-03	3
	2021-10-05	1
	Texas de la composition della	

--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

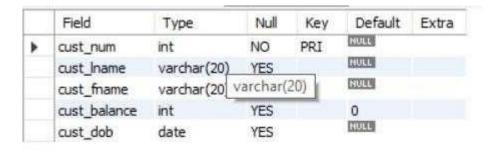
AND CUST\_BALANCE>100000

AND CUST\_NUM NOT IN (SELECT CUST\_NUM FROM GOLD\_MASTER));



--Add a new attribute CUST\_DOB in customer table

ALTER TABLE CUSTOMER ADD COLUMN CUST\_DOB DATE; DESC CUSTOMER;



# **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	<b>→</b>		~	Department_ID	Name	Location_ID
	Edit	<b>З</b> Сору	Delete	1	0	302
	Edit	<b>З</b> Сору	Delete	2	0	706
	Edit	<b>З</b> Сору	Delete	3	0	890
	Edit	<b>≩</b> сору	Delete	4	0	6509
	Edit	<b>≩</b> сору	Delete	5	0	651
	Edit	<b>≩</b> сору	Delete	6	0	471
	Edit	<b>≩</b> сору	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);



# **QUERIES**

a). SELECT COUNT(Employee\_ID)

FROM Employee

WHERE hire\_date> '2015-03-01' And hire\_date<'2015-04-01';



b). SELECT \*

FROM employee

**GROUP BY salary** 

ORDER BY salary DESC LIMIT 1;



	MENT_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;



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;