DEPARTMENT OF COMPUTER SCIENCE AND APPLICATIONS
PANJAB UNIVERSITY,
CHANDIGARH

SQL PROJECT

TOPIC: FRAUD DETECTION USING SQL

SUBMITTED BY: AMISHA SHARMA

COURSE: MCA (SELF FINANCED)-I

ROLL NUMBER: 56

SUBMITTED TO: DR. KAVITA TANEJA

FOR THE TOPIC “FRAUD DETECTION USING SQL” WE WILL USE THREE TABLES – ONE FOR THE USERS NAMED AS ‘USERS’ , SECOND FOR THE TRANSACTIONS MADE BY USERS WHICH IS NAMED AS ‘TRANSACTIONS’ AND LAST FOR THE FRAUD REPORTS REPORTED BY THE USERS WHICH IS NAMED AS ‘FRAUD_REPORTS’.

1. CREATE TABLE

HERE, WE WILL CREATE THE REQUIRED THREE TABLES USING CREATE TABLE COMMAND.

(A) USERS TABLE

```sql
CREATE TABLE users (
    user_id INT PRIMARY KEY,
    username VARCHAR(100) NOT NULL,
    email VARCHAR(100) NOT NULL UNIQUE,
    created_at DATETIME NOT NULL,
    last_login DATETIME
);
```

(B) TRANSACTIONS TABLE

```sql
CREATE TABLE transactions (
    transaction_id INT PRIMARY KEY,
    user_id INT,
    transaction_date DATETIME,
    amount DECIMAL(10, 2),
    transaction_type VARCHAR(40),
    status VARCHAR(40),
    FOREIGN KEY (user_id) REFERENCES users(user_id)
);
```

(C) FRAUD REPORTS TABLE

```sql
CREATE TABLE fraud_reports (
    report_id INT PRIMARY KEY,
    transaction_id INT,
    report_date DATETIME,
    reported_by VARCHAR(100),
    status VARCHAR(40),
    FOREIGN KEY (transaction_id) REFERENCES transactions(transaction_id)
);
```

2. INSERTING VALUES INTO TABLES

(A) USERS TABLE

```sql
INSERT INTO users (user_id, username, email, created_at, last_login) VALUES
(1, 'Aarav Sharma', 'aarav.sharma@email.com', '2024-01-01 10:00:00', '2024-11-01 09:00:00'),
(2, 'Vivaan Gupta', 'vivaan.gupta@email.com', '2024-01-02 11:00:00', '2024-11-02 10:00:00'),
(3, 'Aditya Verma', 'aditya.verma@email.com', '2024-01-03 12:00:00', '2024-11-03 11:00:00'),
(4, 'Vihaan Reddy', 'vihaan.reddy@email.com', '2024-01-04 13:00:00', '2024-11-04 12:00:00'),
(5, 'Reyansh Patel', 'reyansh.patel@email.com', '2024-01-05 14:00:00', '2024-11-05 13:00:00'),
(6, 'Krishna Iyer', 'krishna.iyer@email.com', '2024-01-06 15:00:00', '2024-11-06 14:00:00'),
(7, 'Sai Kiran', 'sai.kiran@email.com', '2024-01-07 16:00:00', '2024-11-07 15:00:00'),
(8, 'Anaya Joshi', 'anaya.joshi@email.com', '2024-01-08 17:00:00', '2024-11-08 16:00:00'),
(9, 'Diya Mehta', 'diya.mehta@email.com', '2024-01-09 18:00:00', '2024-11-09 17:00:00'),
(10, 'Pooja Nair', 'pooja.nair@email.com', '2024-01-10 19:00:00', '2024-11-10 18:00:00'),
(11, 'Rohan Singh', 'rohan.singh@email.com', '2024-01-11 20:00:00', '2024-11-11 19:00:00'),
(12, 'Sneha Bhatia', 'sneha.bhatia@email.com', '2024-01-12 21:00:00', '2024-11-12 20:00:00'),
(13, 'Karan Kapoor', 'karan.kapoor@email.com', '2024-01-13 22:00:00', '2024-11-13 21:00:00'),
(14, 'Nisha Agarwal', 'nisha.agarwal@email.com', '2024-01-14 23:00:00', '2024-11-14 22:00:00'),
(15, 'Riya Choudhary', 'riya.choudhary@email.com', '2024-01-15 00:00:00', '2024-11-15 23:00:00'),
(16, 'Aditi S harma', 'aditi.sharma@email.com', '2024-01-16 01:00:00', '2024-11-16 00:00:00'),
(17, 'Tanvi Desai', 'tanvi.desai@email.com', '2024-01-17 02:00:00', '2024-11-17 01:00:00'),
(18, 'Kavya Rani', 'kavya.rani@email.com', '2024-01-18 03:00:00', '2024-11-18 02:00:00'),
(19, 'Ayaan Khan', 'ayaan.khan@email.com', '2024-01-19 04:00:00', '2024-11-19 03:00:00'),
(20, 'Shivam Yadav', 'shivam.yadav@email.com', '2024-01-20 05:00:00', '2024-11-20 04:00:00');
```

(B) TRANSACTIONS TABLE

```sql
INSERT INTO transactions (transaction_id, user_id, transaction_date, amount,
transaction_type, status) VALUES
(1, 1, '2024-10-01 10:30:00', 1000.00, 'credit', 'completed'),
(2, 1, '2024-10-02 11:30:00', 500.00, 'debit', 'completed'),
(3, 2, '2024-10-03 12:30:00', 2000.00, 'credit', 'completed'),
(4, 2, '2024-10-04 13:30:00', 1500.00, 'debit', 'pending'),
(5, 3, '2024-10-05 14:30:00', 3000.00, 'credit', 'failed'),
(6, 3, '2024-10-06 15:30:00', 2500.00, 'debit', 'completed'),
(7, 4, '2024-10-07 16:30:00', 4000.00, 'credit', 'completed'),
(8, 4, '2024-10-08 17:30:00', 3500.00, 'debit', 'completed'),
(9, 5, '2024-10-09 18:30:00', 5000.00, 'credit', 'completed'),
(10, 5, '2024-10-10 19:30:00', 4500.00, 'debit', 'under_investigation'),
(11, 6, '2024-10-11 20:30:00', 6000.00, 'credit', 'completed'),
(12, 6, '2024-10-12 21:30:00', 5500.00, 'debit', 'completed'),
(13, 7, '2024-10-13 22:30:00', 7000.00, 'credit', 'completed'),
(14, 7, '2024-10-14 23:30:00', 6500.00, 'debit', 'completed'),
(15, 8, '2024-10-15 00:30:00', 8000.00, 'credit', 'completed'),
(16, 8, '2024-10-16 01:30:00', 7500.00, 'debit', 'completed'),
(17, 9, '2024-10-17 02:30:00', 9000.00, 'credit', 'completed'),
(18, 9, '2024-10-18 03:30:00', 8500.00, 'debit', 'completed'),
(19, 10, '2024-10-19 04:30:00', 10000.00, 'credit', 'completed'),
(20, 10, '2024-10-20 05:30:00', 9500.00, 'debit', 'completed');
```

(C) FRAUD REPORTS TABLES

```
INSERT INTO fraud_reports (report_id, transaction_id, report_date, reported_by, status) VALUES
(1, 10, '2024-10-21 06:00:00', 'admin', 'under_investigation'),
(2, 15, '2024-10-22 07:00:00', 'user1', 'resolved'),
(3, 5, '2024-10-23 08:00:00', 'user2', 'dismissed'),
(4, 12, '2024-10-24 09:00:00', 'admin', 'under_investigation'),
(5, 18, '2024-10-25 10:00:00', 'user3', 'resolved'),
(6, 3, '2024-10-26 11:00:00', 'user4', 'dismissed'),
(7, 7, '2024-10-27 12:00:00', 'admin', 'under_investigation'),
(8, 14, '2024-10-28 13:00:00', 'user5', 'resolved'),
(9, 1, '2024-10-29 14:00:00', 'user6', 'dismissed'),
(10, 9, '2024-10-30 15:00:00', 'admin', 'under_investigation'),
(11, 11, '2024-10-31 16:00:00', 'user7', 'resolved'),
(12, 17, '2024-11-01 17:00:00', 'user8', 'dismissed'),
(13, 4, '2024-11-02 18:00:00', 'admin', 'under_investigation'),
(14, 8, '2024-11-03 19:00:00', 'user9', 'resolved'),
(15, 19, '2024-11-04 20:00:00', 'user10', 'dismissed'),
(16, 2, '2024-11-05 21:00:00', 'admin', 'under_investigation'),
(17, 6, '2024-11-06 22:00:00', 'user11', 'resolved'),
(18, 13, '2024-11-07 23:00:00', 'user12', 'dismissed'),
(19, 16, '2024-11-08 00:00:00', 'admin', 'under_investigation'),
(20, 20, '2024-11-09 01:00:00', 'user13', 'resolved');
```

3. SELECT COMMAND

(A) USERS TABLES

```
SELECT * FROM users;
```

OUTPUT IS:

user_id	username	email	created_at	last_login
1	Aarav Sharma	aarav.sharma@email.com	2024-01-01 10:00:00	2024-11-01 09:00:00
2	Vivaan Gupta	vivaan.gupta@email.com	2024-01-02 11:00:00	2024-11-02 10:00:00
3	Aditya Verma	aditya.verma@email.com	2024-01-03 12:00:00	2024-11-03 11:00:00
4	Vihaan Reddy	vihaan.reddy@email.com	2024-01-04 13:00:00	2024-11-04 12:00:00
5	Reyansh Patel	reyansh.patel@email.com	2024-01-05 14:00:00	2024-11-05 13:00:00
6	Krishna Iyer	krishna.iyer@email.com	2024-01-06 15:00:00	2024-11-06 14:00:00
7	Sai Kiran	sai.kiran@email.com	2024-01-07 16:00:00	2024-11-07 15:00:00
8	Anaya Joshi	anaya.joshi@email.com	2024-01-08 17:00:00	2024-11-08 16:00:00
9	Diya Mehta	diya.mehta@email.com	2024-01-09 18:00:00	2024-11-09 17:00:00
10	Pooja Nair	pooja.nair@email.com	2024-01-10 19:00:00	2024-11-10 18:00:00
11	Rohan Singh	rohan.singh@email.com	2024-01-11 20:00:00	2024-11-11 19:00:00
12	Sneha Bhatia	sneha.bhatia@email.com	2024-01-12 21:00:00	2024-11-12 20:00:00
13	Karan Kapoor	karan.kapoor@email.com	2024-01-13 22:00:00	2024-11-13 21:00:00
14	Nisha Agarwal	nisha.agarwal@email.com	2024-01-14 23:00:00	2024-11-14 22:00:00
15	Riya Choudhary	riya.choudhary@email.com	2024-01-15 00:00:00	2024-11-15 23:00:00
16	Aditi S harma	aditi.sharma@email.com	2024-01-16 01:00:00	2024-11-16 00:00:00
17	Tanvi Desai	tanvi.desai@email.com	2024-01-17 02:00:00	2024-11-17 01:00:00
18	Kavya Rani	kavya.rani@email.com	2024-01-18 03:00:00	2024-11-18 02:00:00
19	Ayaan Khan	ayaan.khan@email.com	2024-01-19 04:00:00	2024-11-19 03:00:00
20	Shivam Yadav	shivam.yadav@email.com	2024-01-20 05:00:00	2024-11-20 04:00:00

(B) TRANSACTIONS TABLE

```
SELECT * FROM TRANSACTIONS;
```

OUTPUT IS:

transaction_id	user_id	transaction_date	amount	transaction_type	status
1	1	2024-10-01 10:30...	1000	credit	completed
2	1	2024-10-02 11:30...	500	debit	completed
3	2	2024-10-03 12:30...	2000	credit	completed
4	2	2024-10-04 13:30...	1500	debit	pending
5	3	2024-10-05 14:30...	3000	credit	failed
6	3	2024-10-06 15:30...	2500	debit	completed
7	4	2024-10-07 16:30...	4000	credit	completed
8	4	2024-10-08 17:30...	3500	debit	completed
9	5	2024-10-09 18:30...	5000	credit	completed
10	5	2024-10-10 19:30...	4500	debit	under_investig
11	6	2024-10-11 20:30...	6000	credit	completed
12	6	2024-10-12 21:30...	5500	debit	completed
13	7	2024-10-13 22:30...	7000	credit	completed
14	7	2024-10-14 23:30...	6500	debit	completed
15	8	2024-10-15 00:30...	8000	credit	completed
16	8	2024-10-16 01:30...	7500	debit	completed
17	9	2024-10-17 02:30...	9000	credit	completed
18	9	2024-10-18 03:30...	8500	debit	completed
19	10	2024-10-19 04:30...	10000	credit	completed
20	10	2024-10-20 05:30...	9500	debit	completed

(C) FRAUD REPORTS TABLE

```
SELECT* FROM FRAUD_REPORTS;
```

OUTPUT IS:

report_id	transaction_id	report_date	reported_by	status
1	10	2024-10-21 06:00:00	admin	under_investigation
2	15	2024-10-22 07:00:00	user1	resolved
3	5	2024-10-23 08:00:00	user2	dismissed
4	12	2024-10-24 09:00:00	admin	under_investigation
5	18	2024-10-25 10:00:00	user3	resolved
6	3	2024-10-26 11:00:00	user4	dismissed
7	7	2024-10-27 12:00:00	admin	under_investigation
8	14	2024-10-28 13:00:00	user5	resolved
9	1	2024-10-29 14:00:00	user6	dismissed
10	9	2024-10-30 15:00:00	admin	under_investigation
11	11	2024-10-31 16:00:00	user7	resolved
12	17	2024-11-01 17:00:00	user8	dismissed
13	4	2024-11-02 18:00:00	admin	under_investigation
14	8	2024-11-03 19:00:00	user9	resolved
15	19	2024-11-04 20:00:00	user10	dismissed
16	2	2024-11-05 21:00:00	admin	under_investigation
17	6	2024-11-06 22:00:00	user11	resolved
18	13	2024-11-07 23:00:00	user12	dismissed
19	16	2024-11-08 00:00:00	admin	under_investigation
20	20	2024-11-09 01:00:00	user13	resolved

4. QUERY DATABASE TABLE

IT IS DONE WITH THE HELP OF SELECT COMMAND WHICH GIVES THE USERS WHO HAVE REPORTED FRAUD:

```
SELECT DISTINCT reported_by FROM fraud_reports;
```

OUTPUT IS:

reported_by
admin
user1
user2
user3
user4
user5
user6
user7
user8
user9
user10
user11
user12
user13

5. CONDITIONAL RETRIEVEL OF ROWS

IT CAN BE DONE BY EXECUTING SELECT COMMAND ALONG WITH THE WHERE CLAUSE. HERE, WE WILL GET THE TRANSACTIONS BETWEEN TWO DATES AS:

```
SELECT * FROM transactions
WHERE transaction_date BETWEEN '2024-10-01' AND '2024-10-31';
```

OUTPUT IS:

transaction_id	user_id	transaction_date	amount	transaction_type	status
1	1	2024-10-01 10:30:00	1000	credit	completed
2	1	2024-10-02 11:30:00	500	debit	completed
3	2	2024-10-03 12:30:00	2000	credit	completed
4	2	2024-10-04 13:30:00	1500	debit	pending
5	3	2024-10-05 14:30:00	3000	credit	failed
6	3	2024-10-06 15:30:00	2500	debit	completed
7	4	2024-10-07 16:30:00	4000	credit	completed
8	4	2024-10-08 17:30:00	3500	debit	completed
9	5	2024-10-09 18:30:00	5000	credit	completed
10	5	2024-10-10 19:30:00	4500	debit	under_investigation
11	6	2024-10-11 20:30:00	6000	credit	completed
12	6	2024-10-12 21:30:00	5500	debit	completed

6. **Queries for Working with NULL Values**

IT IS USED WITH THE HELP OF IS NULL CONDITION AND HERE IT HELPS IN RETRIEVING THE REPORTS THAT ARE NOT REPORTED AS FRAUD BY THE USER WHICH IS AS FOLLOWS:

```
SELECT * FROM fraud_reports WHERE reported_by IS NULL;
```

OUTPUT IS:

report_id	transaction_id	report_date	reported_by	status

THE OUTPUT GIVES NO RESULT BECAUSE NOT NULL CONSTRAINT IS APPLIED ON THE REPORTED_BY COLUMN OF THE TABLE.

7. **Queries for Pattern Matching**

IT CAN BE DONE BY USING THE LIKE OPERATOR. HERE, WE WILL GET USERS WITH A USERNAME CONTAINING A SPECIFIC SUBSTRING AS FOLLOWS:

```
SELECT * FROM users WHERE username LIKE '%an%';
```

OUTPUT IS:

user_id	username	email	created_at	last_login
2	Vivaan Gupta	vivaan.gupta@email.com	2024-01-02 11:00:00	2024-11-02 10:00:00
4	Vihaan Reddy	vihaan.reddy@email.com	2024-01-04 13:00:00	2024-11-04 12:00:00
5	Reyansh Patel	reyansh.patel@email.com	2024-01-05 14:00:00	2024-11-05 13:00:00
7	Sai Kiran	sai.kiran@email.com	2024-01-07 16:00:00	2024-11-07 15:00:00
8	Anaya Joshi	anaya.joshi@email.com	2024-01-08 17:00:00	2024-11-08 16:00:00
11	Rohan Singh	rohan.singh@email.com	2024-01-11 20:00:00	2024-11-11 19:00:00
13	Karan Kapoor	karan.kapoor@email.com	2024-01-13 22:00:00	2024-11-13 21:00:00
17	Tanvi Desai	tanvi.desai@email.com	2024-01-17 02:00:00	2024-11-17 01:00:00
18	Kavya Rani	kavya.rani@email.com	2024-01-18 03:00:00	2024-11-18 02:00:00
19	Ayaan Khan	ayaan.khan@email.com	2024-01-19 04:00:00	2024-11-19 03:00:00

8. **Queries for Ordering Results**

THESE CAN BE ACHIEVED BY USING ORDER BY CLAUSE WHICH HELPS IN SEQUENCING RESULT IN ASCENDING OR DESECENDING ORDER. HERE, WE WILL GET USERS ORDERED BY CREATED DATE AND LAST LOGIN AS FOLLOWS:

```
SELECT * FROM users ORDER BY created_at ASC, last_login DESC;
```

OUTPUT IS:

user_id	username	email	created_at	last_login
1	Aarav Sharma	aarav.sharma@email.com	2024-01-01 10:00:00	2024-11-01 09:00:00
2	Vivaan Gupta	vivaan.gupta@email.com	2024-01-02 11:00:00	2024-11-02 10:00:00
3	Aditya Verma	aditya.verma@email.com	2024-01-03 12:00:00	2024-11-03 11:00:00
4	Vihaan Reddy	vihaan.reddy@email.com	2024-01-04 13:00:00	2024-11-04 12:00:00
5	Reyansh Patel	reyansh.patel@email.com	2024-01-05 14:00:00	2024-11-05 13:00:00
6	Krishna Iyer	krishna.iyer@email.com	2024-01-06 15:00:00	2024-11-06 14:00:00
7	Sai Kiran	sai.kiran@email.com	2024-01-07 16:00:00	2024-11-07 15:00:00
8	Anaya Joshi	anaya.joshi@email.com	2024-01-08 17:00:00	2024-11-08 16:00:00
9	Diya Mehta	diya.mehta@email.com	2024-01-09 18:00:00	2024-11-09 17:00:00
10	Pooja Nair	pooja.nair@email.com	2024-01-10 19:00:00	2024-11-10 18:00:00
11	Rohan Singh	rohan.singh@email.com	2024-01-11 20:00:00	2024-11-11 19:00:00
12	Sneha Bhatia	sneha.bhatia@email.com	2024-01-12 21:00:00	2024-11-12 20:00:00

9. **Queries Using Aggregate Functions**

THE AGGREGATE FUNCTIONS COUNT (), SUM (), AVG (), MIN () AND MAX () ARE USED HERE TO GET VARIOUS OUTPUTS AS FOLLOWS:

(i) COUNT ()

```
SELECT status, COUNT(*) AS report_count
FROM fraud_reports
GROUP BY status;
```

OUTPUT IS:

status	report_count
dismissed	6
resolved	7
under_investigation	7

(ii) SUM ()

```
SELECT user_id, SUM(amount) AS total_amount
FROM transactions
GROUP BY user_id;
```

OUTPUT IS:

user_id	total_amount
1	1500
2	3500
3	5500
4	7500
5	9500
6	11500
7	13500
8	15500
9	17500
10	19500

(iii) AVG ()

```
SELECT AVG(amount) AS average_successful_transaction_amount
FROM transactions
WHERE status = 'completed';
```

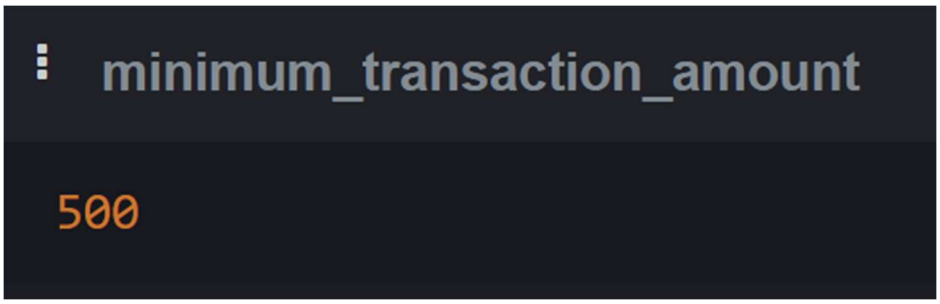
OUTPUT IS:

average_successful_transaction_amount
5647.058823529412

(iv) MIN ()

```
SELECT MIN(amount) AS minimum_transaction_amount FROM transactions;
```

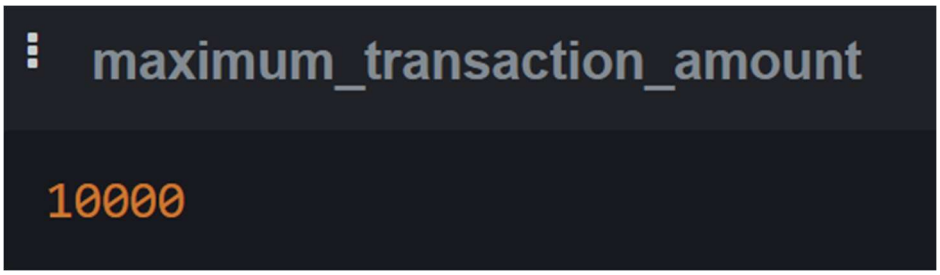
OUTPUT IS:



(v) MAX ()

```
SELECT MAX(amount) AS maximum_transaction_amount FROM transactions;
```

OUTPUT IS:



10. **Queries for Grouping Results**

IN THIS, WE WILL USE GROUP BY CLAUSE WHICH WILL GIVE COUNT OF NUMBER OF USERS WITH EACH LOGGED IN ON EACH DATE. THIS IS AS FOLLOWS:

```
SELECT DATE(last_login) AS last_login_date, COUNT(*) AS user_count
FROM users
GROUP BY DATE(last_login);
```

OUTPUT IS:

last_login_date	user_count
2024-11-01	1
2024-11-02	1
2024-11-03	1
2024-11-04	1
2024-11-05	1
2024-11-06	1
2024-11-07	1
2024-11-08	1
2024-11-09	1
2024-11-10	1
2024-11-11	1
2024-11-12	1
2024-11-13	1
2024-11-14	1
2024-11-15	1
2024-11-16	1
2024-11-17	1
2024-11-18	1
2024-11-19	1
2024-11-20	1

11. **Query for Inner Join**

HERE, WE WILL USE INNER JOIN OPERATION TO RETRIEVE SOME COLUMNS OR FIELDS FROM THE USERS TABLE, WHICH IS AS FOLLOWS:

```sql
SELECT u.user_id, u.username, t.transaction_id, t.amount
FROM users u
INNER JOIN transactions t ON u.user_id = t.user_id;
```

OUTPUT IS:

user_id	username	transaction_id	amount
1	Aarav Sharma	1	1000
1	Aarav Sharma	2	500
2	Vivaan Gupta	3	2000
2	Vivaan Gupta	4	1500
3	Aditya Verma	5	3000
3	Aditya Verma	6	2500
4	Vihaan Reddy	7	4000
4	Vihaan Reddy	8	3500
5	Reyansh Patel	9	5000
5	Reyansh Patel	10	4500
6	Krishna Iyer	11	6000
6	Krishna Iyer	12	5500
7	Sai Kiran	13	7000
7	Sai Kiran	14	6500
8	Anaya Joshi	15	8000
8	Anaya Joshi	16	7500
9	Diya Mehta	17	9000
9	Diya Mehta	18	8500
10	Pooja Nair	19	10000
10	Pooia Nair	20	9500

12. **Query for Left Join**

HERE, WE ARE USING LEFT JOIN TO RETRIEVE ALL USERS AND THEIR TRANSACTIONS AS FOLLOWS:

```
SELECT u.user_id, u.username, t.transaction_id, t.amount
FROM users u
LEFT JOIN transactions t ON u.user_id = t.user_id;
```

OUTPUT IS:

user_id	username	transaction_id	amount
1	Aarav Sharma	1	1000
1	Aarav Sharma	2	500
2	Vivaan Gupta	3	2000
2	Vivaan Gupta	4	1500
3	Aditya Verma	5	3000
3	Aditya Verma	6	2500
4	Vihaan Reddy	7	4000
4	Vihaan Reddy	8	3500
5	Reyansh Patel	9	5000
5	Reyansh Patel	10	4500
6	Krishna Iyer	11	6000
6	Krishna Iyer	12	5500
7	Sai Kiran	13	7000
7	Sai Kiran	14	6500
8	Anaya Joshi	15	8000
8	Anaya Joshi	16	7500
9	Diya Mehta	17	9000
9	Diya Mehta	18	8500
10	Pooja Nair	19	10000
10	Pooja Nair	20	9500
11	Rohan Singh	NULL	NULL
12	Sneha Bhatia	NULL	NULL
13	Karan Kapoor	NULL	NULL
14	Nisha Agarwal	NULL	NULL
15	Riya Choudhary	NULL	NULL
16	Aditi S harma	NULL	NULL
17	Tanvi Desai	NULL	NULL
18	Kavya Rani	NULL	NULL
19	Ayaan Khan	NULL	NULL
20	Shivam Yadav	NULL	NULL

13. **Query for Right Join**

HERE, WE WILL USE RIGHT JOIN OPERATION TO RETRIEVE ALL TRANSACTIONS AND THEIR ASSOCIATED USERS AS FOLLOWS:

```sql
SELECT u.user_id, u.username, t.transaction_id, t.amount
FROM users u
RIGHT JOIN transactions t ON u.user_id = t.user_id;
```

OUTPUT IS:

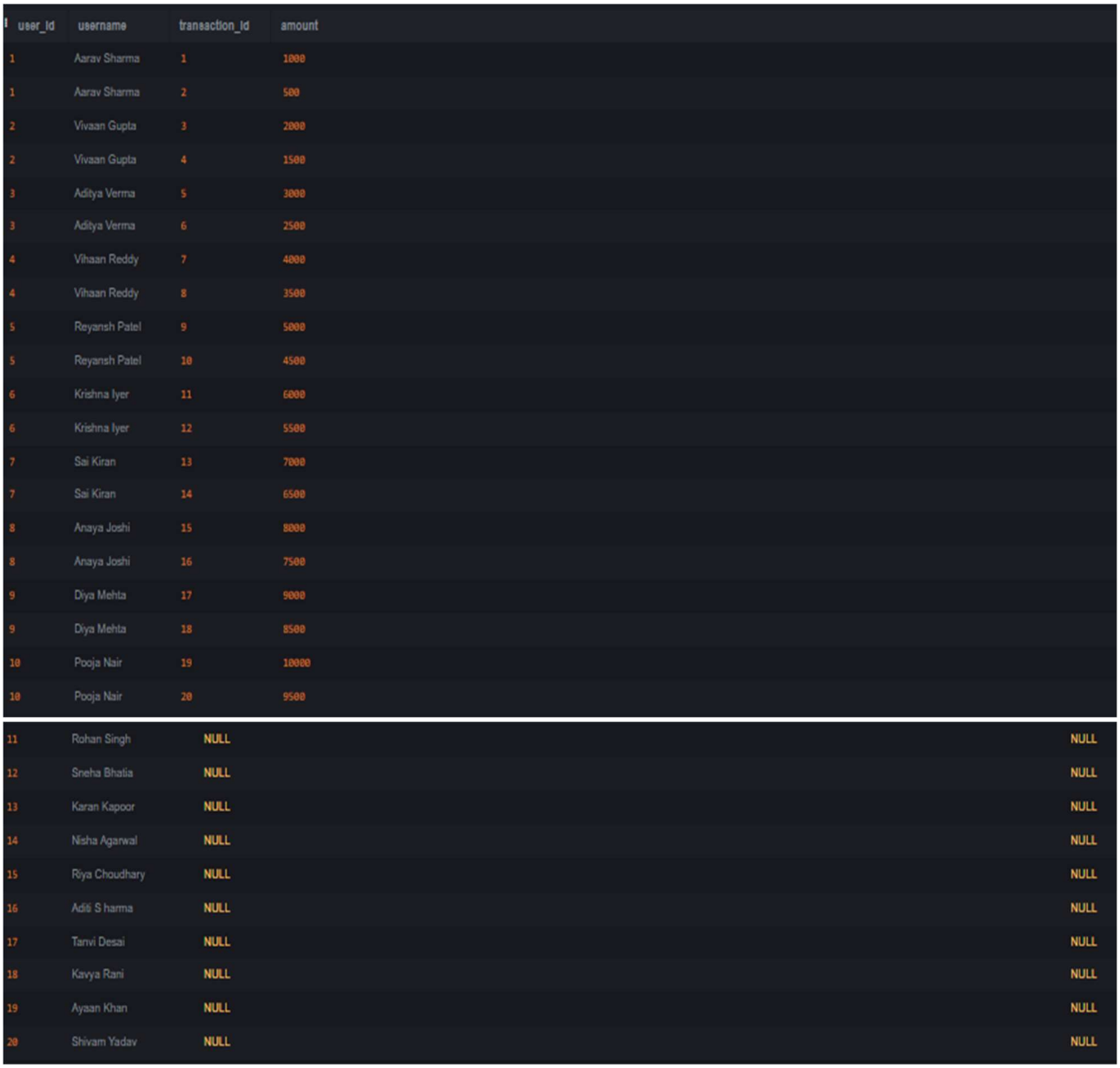
user_id	username	transaction_id	amount
1	Aarav Sharma	1	1000
1	Aarav Sharma	2	500
2	Vivaan Gupta	3	2000
2	Vivaan Gupta	4	1500
3	Aditya Verma	5	3000
3	Aditya Verma	6	2500
4	Vihaan Reddy	7	4000
4	Vihaan Reddy	8	3500
5	Reyansh Patel	9	5000
5	Reyansh Patel	10	4500
6	Krishna Iyer	11	6000
6	Krishna Iyer	12	5500
7	Sai Kiran	13	7000
7	Sai Kiran	14	6500
8	Anaya Joshi	15	8000
8	Anaya Joshi	16	7500
9	Diya Mehta	17	9000
9	Diya Mehta	18	8500
10	Pooja Nair	19	10000
10	Pooja Nair	20	9500

14. Query for Full Outer Join

HERE WE WILL USE FULL OUTER JOIN WHICH HELPS IN RETRIEVING ALL THE RECORDS FROM TWO TABLES AS FOLLOWS:

```
SELECT u.user_id, u.username, t.transaction_id, t.amount
FROM users u
FULL OUTER JOIN transactions t ON u.user_id = t.user_id;
```

OUTPUT IS:



user_id	username	transaction_id	amount
1	Aarav Sharma	1	1000
1	Aarav Sharma	2	500
2	Vivaan Gupta	3	2000
2	Vivaan Gupta	4	1500
3	Aditya Verma	5	3000
3	Aditya Verma	6	2500
4	Vihaan Reddy	7	4000
4	Vihaan Reddy	8	3500
5	Reyansh Patel	9	5000
5	Reyansh Patel	10	4500
6	Krishna Iyer	11	6000
6	Krishna Iyer	12	5500
7	Sai Kiran	13	7000
7	Sai Kiran	14	6500
8	Anaya Joshi	15	8000
8	Anaya Joshi	16	7500
9	Diya Mehta	17	9000
9	Diya Mehta	18	8500
10	Pooja Nair	19	10000
10	Pooja Nair	20	9500
11	Rohan Singh	NULL	NULL
12	Sneha Bhatia	NULL	NULL
13	Karan Kapoor	NULL	NULL
14	Nisha Agarwal	NULL	NULL
15	Riya Choudhary	NULL	NULL
16	Aditi S harma	NULL	NULL
17	Tanvi Desai	NULL	NULL
18	Kavya Rani	NULL	NULL
19	Ayaan Khan	NULL	NULL
20	Shivam Yadav	NULL	NULL

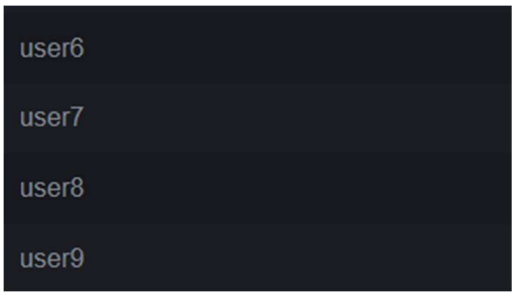
15.**Query for UNION**

IN THIS CASE, **UNION** operator retrieves a combined list of unique usernames from the **users** table and the names of users who reported fraud from the **fraud_reports** table AS FOLLOWS:

```
SELECT username AS name FROM users
UNION
SELECT reported_by AS name FROM fraud_reports;
```

OUTPUT IS:

name
Aarav Sharma
Aditi S harma
Aditya Verma
Anaya Joshi
Ayaan Khan
Diya Mehta
Karan Kapoor
Kavya Rani
Krishna Iyer
Nisha Agarwal

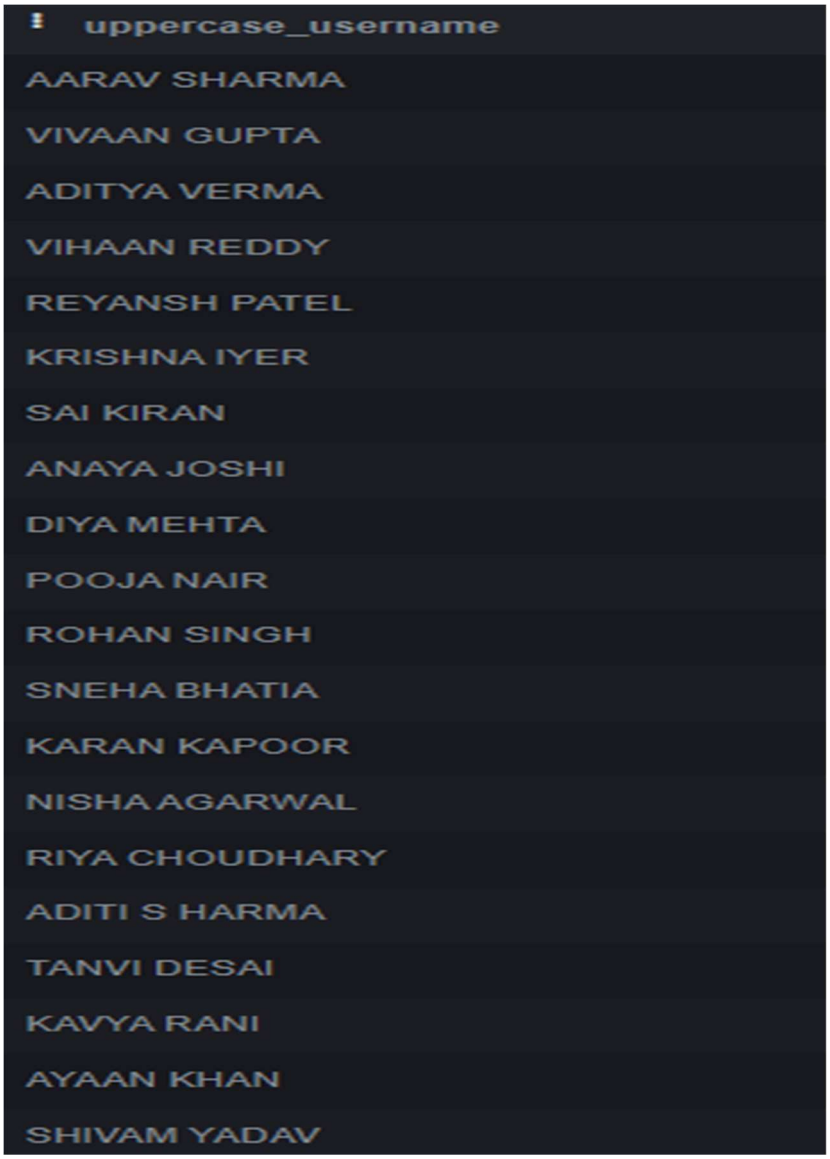


16. **Queries for Character Functions**

IN THIS CASE, CHARACTER FUNCTION IS USED TO GET THE NAMES IN UPPER CASE AS FOLLOWS:

```
 SELECT UPPER(username) AS uppercase_username
FROM users;
```

OUTPUT IS:



17. CREATING a trigger

THIS COMMAND creates a trigger AS FOLLOWS:

```
CREATE TABLE user_log (
    log_id INTEGER PRIMARY KEY,
    user_id INTEGER,
    ACTION TEXT,
    action_time DATETIME DEFAULT CURRENT_TIMESTAMP
);

CREATE TRIGGER after_user_insert
AFTER INSERT ON users
BEGIN
    INSERT INTO user_log (user_id, ACTION) VALUES (new.user_id, 'Inserted');
END;
```

NOW TO ADD SOME DATA IN THE CREATED TRIGGER WE CAN DO THE FOLLOWING STEP:

```
INSERT INTO user_log  ( log_id , user_id ,ACTION, action_time )
VALUES (1, 100, 'inserted', '2024-05-03 12:00:00');
```

TO SEE THE RESULT OF THIS OPERATION WE WILL USE SELECT STATEMENT AS:

```
SELECT * FROM user_log;
```

OUTPUT IS:

log_id	user_id	action	action_time
1	100	inserted	2024-05-03 12:00:00

18. Trigger to Prevent Deletion from USERS Table

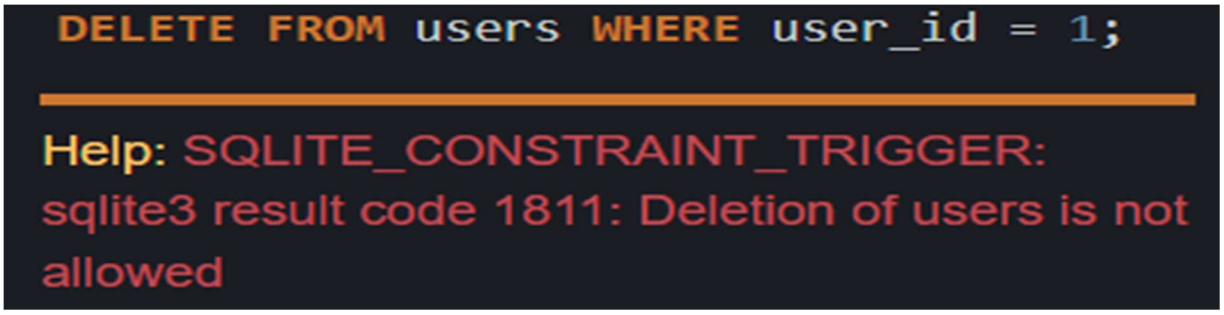
TO DO THIS, WE WILL PERFORM THE FOLLOWING:

```
CREATE TRIGGER prevent_user_delete
BEFORE DELETE ON users
BEGIN
    INSERT INTO user_log (user_id, ACTION) VALUES (old.user_id, 'Attempted to delete user');
    SELECT RAISE(ABORT, 'Deletion of users is not allowed');
END;
```

NOW, WE WILL CHECK IF THIS COMMAND WORKS AS FOLLOWS:

```
DELETE FROM users WHERE user_id = 1;
```

OUTPUT OF THIS WILL BE:



HERE, THE OUTPUT SHOWS THE ERROR WHICH DOES NOT ALLOWS THE DELETION OPERATION FROM USERS TABLE.

19. Query Using a Subquery

THIS IS EXECUTED AS FOLLOWS:

```sql
SELECT username
FROM users
WHERE user_id IN (SELECT user_id FROM transactions WHERE amount > 5000);
```

OUTPUT IS:

username
Krishna Iyer
Sai Kiran
Anaya Joshi
Diya Mehta
Pooja Nair

20.UPDATE QUERY

THIS COMMAND IS IMPLEMENTED AS:

```sql
UPDATE transactions
SET amount = 6000.00, status = 'completed'
WHERE transaction_id = 11;
```

OUTPUT IS:

transaction_id	user_id	transaction_date	amount	transaction_type	status
1	1	2024-10-01 10:30:00	1000	credit	completed
2	1	2024-10-02 11:30:00	500	debit	completed
3	2	2024-10-03 12:30:00	2000	credit	completed
4	2	2024-10-04 13:30:00	1500	debit	pending
5	3	2024-10-05 14:30:00	3000	credit	failed
6	3	2024-10-06 15:30:00	2500	debit	completed
7	4	2024-10-07 16:30:00	4000	credit	completed
8	4	2024-10-08 17:30:00	3500	debit	completed
9	5	2024-10-09 18:30:00	5000	credit	completed
10	5	2024-10-10 19:30:00	4500	debit	under_investigation
11	6	2024-10-11 20:30:00	6000	credit	completed
12	6	2024-10-12 21:30:00	5500	debit	completed
13	7	2024-10-13 22:30:00	7000	credit	completed
14	7	2024-10-14 23:30:00	6500	debit	completed
15	8	2024-10-15 00:30:00	8000	credit	completed
16	8	2024-10-16 01:30:00	7500	debit	completed
17	9	2024-10-17 02:30:00	9000	credit	completed
18	9	2024-10-18 03:30:00	8500	debit	completed
19	10	2024-10-19 04:30:00	10000	credit	completed
20	10	2024-10-20 05:30:00	9500	debit	completed