Milaap Assignment

By Abhigyan Dutta

Creating the tables

```
CREATE TABLE campaigns (
   id SERIAL PRIMARY KEY,
   project_id INT NOT NULL,
   name VARCHAR(1000) NOT NULL,
   goal_amount NUMERIC(12, 2) NOT NULL,
   submitted_on DATE NOT NULL,
   channel VARCHAR(50) NOT NULL
);
-- Inserting data into the table
INSERT INTO campaigns (id, project_id, name, goal_amount, submitted_on, channel)
VALUES
   (1, 2, 'A', 200000, '2023-01-07', 'direct'),
   (2, 3, 'B', 300000, '2024-01-03', 'google'),
   (3, 4, 'C', 400000, '2024-01-05', 'google'),
   (4, 2, 'D', 100000, '2023-01-10', 'facebook'),
   (5, 5, 'E', 2500000, '2024-01-09', 'direct'),
   (6, 6, 'F', 450000, '2023-01-12', 'google');

SELECT* from campaigns
```

Output:

Table 1: campaigns

	id [PK] integer	project_id integer	name character varying (1000)	goal_amount numeric (12,2)	submitted_on date	channel character varying (50)
1	1	2	A	200000.00	2023-01-07	direct
2	2	3	B	300000.00	2024-01-03	google
3	3	4	C	400000.00	2024-01-05	google
4	4	2	D	100000.00	2023-01-10	facebook
5	5	5	E	2500000.00	2024-01-09	direct
6	6	6	F	450000.00	2023-01-12	google

```
CREATE TABLE projects (
   id SERIAL PRIMARY KEY,
   category VARCHAR(1000) NOT NULL,
   total_pending_amount NUMERIC(12, 2) NOT NULL
);
-- Insert data into the table
INSERT INTO projects (id, category, total_pending_amount)
VALUES
   (1, 'Medical', 2000),
   (2, 'Medical', 1500),
   (3, 'Memorials', 400),
   (4, 'Medical', 300),
   (5, 'Memorials', 2000),
   (6, 'Education', 4000),
   (7, 'Medical', 1200);

SELECT* from projects
```

Output:

Table 2: projects

	id [PK] integer	category character varying (1000)	total_pending_amount numeric (12,2)
1	1	Medical	2000.00
2	2	Medical	1500.00
3	3	Memorials	400.00
4	4	Medical	300.00
5	5	Memorials	2000.00
6	6	Education	4000.00
7	7	Medical	1200.00

```
CREATE TABLE payments (
   id SERIAL PRIMARY KEY,
   campaign_id INT NOT NULL,
   project_id INT NOT NULL,
   currency VARCHAR(10) NOT NULL,
   amount NUMERIC(12, 2) NOT NULL,
   status VARCHAR(50) NOT NULL
);
-- Insert data into the table
INSERT INTO payments (id, campaign_id, project_id, currency, amount, status)
VALUES
   (1, 2, 3, 'usd', 20, 'success'),
   (2, 3, 4, 'inr', 500, 'success'),
   (3, 1, 2, 'inr', 200, 'success'),
   (4, 2, 3, 'usd', 50, 'failed'),
   (5, 4, 2, 'inr', 1000, 'success'),
   (6, 5, 5, 'usd', 75, 'failed'),
   (7, 2, 3, 'inr', 10000, 'success'),
   (8, 1, 2, 'inr', 2000, 'success');

SELECT* from payments
```

Output:

Table 3: payments

	id [PK] integer	campaign_id integer	project_id integer	currency character varying (10)	amount numeric (12,2)	status character varying (50)
1	1	2	3	usd	20.00	success
2	2	3	4	inr	500.00	success
3	3	1	2	inr	200.00	success
4	4	2	3	usd	50.00	failed
5	5	4	2	inr	1000.00	success
6	6	5	5	usd	75.00	failed
7	7	2	3	inr	10000.00	success
8	8	1	2	inr	2000.00	success

```
CREATE TABLE withdrawals (
   id SERIAL PRIMARY KEY,
   source VARCHAR(50) NOT NULL,
   currency VARCHAR(10) NOT NULL,
   amount_requested NUMERIC(12, 2) NOT NULL,
   status VARCHAR(50) NOT NULL,
```

```
    project_id INT NOT NULL
);
-- Insert data into the table
INSERT INTO withdrawals (id, source, currency, amount_requested, status, project_id)
VALUES
    (1, 'web', 'inr', 200, 'transferred', 2),
    (2, 'web', 'usd', 400, 'transferred', 3),
    (3, 'app', 'inr', 200, 'transferred', 5),
    (4, 'web', 'inr', 50, 'rejected', 1),
    (5, 'app', 'usd', 100, 'transferred', 5),
    (6, 'web', 'inr', 300, 'rejected', 2),
    (7, 'app', 'inr', 400, 'transferred', 1);

SELECT* from withdrawals
```

Output:

Table 4: withdrawals

	id [PK] integer	source character varying (50)	currency character varying (10)	amount_requested numeric (12,2)	status character varying (50)	project_id integer
1	1	web	inr	200.00	transferred	2
2	2	web	usd	400.00	transferred	3
3	3	app	inr	200.00	transferred	5
4	4	web	inr	50.00	rejected	1
5	5	app	usd	100.00	transferred	5
6	6	web	inr	300.00	rejected	2
7	7	app	inr	400.00	transferred	1

Questions

1. List of campaigns where the pending amount is greater than 1k, Submitted in this year, sorted with highest pending amount.

```
SELECT
    c.id AS campaign_id,
    c.name AS campaign_name,
    c.goal_amount,
    pr.total_pending_amount
FROM
    campaigns c
JOIN
    projects pr ON c.project_id = pr.id
WHERE
    pr.total_pending_amount > 1000
    AND EXTRACT(YEAR FROM c.submitted_on) = 2024
ORDER BY
    pr.total_pending_amount DESC;
```

Output:

	campaign_id integer	campaign_name character varying (1000)	goal_amount numeric (12,2)	total_pending_amount numeric (12,2)
1	5	E	2500000.00	2000.00

2. Project wise withdrawal, show currency wise raised and transferred.

```
SELECT
   withdrawals.project_id,
   withdrawals.currency,
   SUM(withdrawals.amount_requested) AS total_raised,
   SUM(CASE WHEN withdrawals.status = 'transferred' THEN withdrawals.amount_requested ELSE 0 END) AS
total_transferred
FROM
   withdrawals
GROUP BY
   withdrawals.project_id, withdrawals.currency
ORDER BY
   withdrawals.project_id, withdrawals.currency;
```

Output:

	project_id integer	currency character varying (10)	total_raised numeric	total_transferred numeric
1	1	inr	450.00	400.00
2	2	inr	500.00	200.00
3	3	usd	400.00	400.00
4	5	inr	200.00	200.00
5	5	usd	100.00	100.00

3. % of withdrawals happened from APP.

```
SELECT
   ROUND(
      (SUM(CASE WHEN source = 'app' THEN 1 ELSE 0 END) * 100.0) / COUNT(*), 2
   ) AS percentage_from_app
FROM
   withdrawals;
```

Output:

	percentage_from_app numeric
1	42.86

4. This year total amount that was requested and total amount that got transferred [all in inr equivalent]

```
SELECT
   SUM(amount_requested * CASE
      WHEN currency = 'usd' THEN 80
      ELSE 1
   END) AS total_requested_in_inr,
   SUM(CASE
      WHEN status = 'transferred' THEN amount_requested *
         CASE
            WHEN currency = 'usd' THEN 80
            ELSE 1
         END
      ELSE 0
   END) AS total_transferred_in_inr
FROM
```

```
    withdrawals
WHERE
    EXTRACT(YEAR FROM CURRENT_DATE) = 2024;
```

Output:

	total_requested_in_inr numeric	total_transferred_in_inr numeric
1	41150.00	40800.00

5. Project wise amount raised and failed amount [inr equivalent].

```
SELECT
    project_id,
    SUM(CASE
        WHEN status = 'success' THEN amount *
            CASE
                WHEN currency = 'usd' THEN 80
                ELSE 1
            END
        ELSE 0
    END) AS total_raised_in_inr,
    SUM(CASE
        WHEN status = 'failed' THEN amount *
            CASE
                WHEN currency = 'usd' THEN 80
                ELSE 1
            END
        ELSE 0
    END) AS total_failed_in_inr
FROM
    payments
GROUP BY
    project_id;
```

Output:

	project_id integer	total_raised_in_inr numeric	total_failed_in_inr numeric
1	3	11600.00	4000.00
2	5	0	6000.00
3	4	500.00	0
4	2	3200.00	0

6. List the campaigns which have amount raised more than 80%. [Raised take in inr equivalent].

```
SELECT
    c.id AS campaign_id,
    c.name AS campaign_name,
    c.goal_amount,
    ROUND(SUM(
        CASE
            WHEN p.currency = 'usd' THEN p.amount * 80 -- Convert USD to INR
            WHEN p.currency = 'inr' THEN p.amount
            ELSE 0
        END
```

```
    ), 2) AS total_raised_inr
FROM
    campaigns c
JOIN
    payments p ON c.id = p.campaign_id AND p.status = 'success'
GROUP BY
    c.id, c.name, c.goal_amount
HAVING
    SUM(
        CASE
            WHEN p.currency = 'usd' THEN p.amount * 80
            WHEN p.currency = 'inr' THEN p.amount
            ELSE 0
        END
    ) > 0.8 * c.goal_amount
ORDER BY
    total_raised_inr DESC;
```

Output:

	campaign_id integer	campaign_name character varying (1000)	goal_amount numeric (12,2)	total_raised_inr numeric

No campaigns meet the criteria of raising more than 80% of their goal amount.

Query to find out do they meet the criteria or not.

```
SELECT
    c.id AS campaign_id,
    c.name AS campaign_name,
    c.goal_amount,
    ROUND(0.8 * c.goal_amount, 2) AS eighty_percent_goal,
    ROUND(SUM(
        CASE
            WHEN p.currency = 'usd' THEN p.amount * 80 -- Convert USD to INR
            WHEN p.currency = 'inr' THEN p.amount
            ELSE 0
        END
    ), 2) AS total_raised_in_inr,
    CASE
        WHEN SUM(
            CASE
                WHEN p.currency = 'usd' THEN p.amount * 80
                WHEN p.currency = 'inr' THEN p.amount
                ELSE 0
            END
        ) > 0.8 * c.goal_amount THEN 'Yes'
        ELSE 'No'
    END AS meets_criteria
FROM
    campaigns c
LEFT JOIN
    payments p ON c.id = p.campaign_id AND p.status = 'success'
GROUP BY
    c.id, c.name, c.goal_amount
ORDER BY
```

```
    total_raised_in_inr DESC;
```

Output:

	campaign_id integer	campaign_name character varying (1000)	goal_amount numeric (12,2)	eighty_percent_goal numeric	total_raised_in_inr numeric	meets_criteria text
1	2	B	300000.00	240000.00	11600.00	No
2	1	A	200000.00	160000.00	2200.00	No
3	4	D	100000.00	80000.00	1000.00	No
4	3	C	400000.00	320000.00	500.00	No
5	6	F	450000.00	360000.00	0.00	No
6	5	E	2500000.00	2000000.00	0.00	No

7. Channel wise amount raised this month sorted with highest raise. [Raised take in inr equivalent].

Note: Since it was not specified which month is this month. Considering this month as January, 2024

```
SELECT
    c.channel,
    ROUND(SUM(
        CASE
            WHEN p.currency = 'usd' THEN p.amount * 80 -- Convert USD to INR
            WHEN p.currency = 'inr' THEN p.amount
            ELSE 0
        END
    ), 2) AS total_raised_in_inr
FROM
    campaigns c
JOIN
    payments p ON c.id = p.campaign_id AND p.status = 'success'
WHERE
    EXTRACT(MONTH FROM c.submitted_on) = 1 -- January
    AND EXTRACT(YEAR FROM c.submitted_on) = 2024 -- Year 2024
GROUP BY
    c.channel
ORDER BY
    total_raised_in_inr DESC;
```

Output:

	channel character varying (50)	total_raised_in_inr numeric
1	google	12100.00

Note there were 2 channels direct and google since direct has no success raised so there is only google left out.

8. Month wise payment success rate.

```
SELECT
    EXTRACT(YEAR FROM c.submitted_on) AS year,
    EXTRACT(MONTH FROM c.submitted_on) AS month,
    COUNT(CASE WHEN p.status = 'success' THEN 1 END) * 100.0 / COUNT(*) AS success_rate
FROM
    payments p
JOIN
    campaigns c ON p.campaign_id = c.id
GROUP BY
```

```
    year, month
ORDER BY
    year, month;
```

Output:

	year numeric	month numeric	success_rate numeric
1	2023	1	100.0000000000000000
2	2024	1	60.0000000000000000