SQL Code

```sql
--BRANCHES TABLE
CREATE TABLE branches (
  branch_id SERIAL PRIMARY KEY,
  branch_name VARCHAR(100) NOT NULL,
  ifsc_code VARCHAR(20) UNIQUE NOT NULL,
  city VARCHAR(60),
  state VARCHAR(60),
  manager_name VARCHAR(100),
  opened_on DATE
);

SELECT * FROM branches;


--CUSTOMER TABLE
CREATE TABLE customers (
  customer_id SERIAL PRIMARY KEY,
  full_name VARCHAR(120) NOT NULL,
  email VARCHAR(120) UNIQUE,
  phone VARCHAR(20),
  city VARCHAR(60),
  state VARCHAR(60),
  dob DATE,
  kyc_status VARCHAR(20) CHECK (kyc_status IN ('verified','pending','rejected')),
  created_at DATE DEFAULT CURRENT_DATE
);

SELECT * FROM customers;


--ACCOUNTS TABLE
CREATE TABLE accounts (
  account_id SERIAL PRIMARY KEY,
  customer_id INT REFERENCES customers(customer_id),
  branch_id INT REFERENCES branches(branch_id),
  account_type VARCHAR(30) CHECK (account_type IN ('savings','current','salary','fixed_deposit')),
  opened_on DATE,
  status VARCHAR(20) CHECK (status IN ('active','dormant','closed')),
  balance NUMERIC(14,2) DEFAULT 0,
  currency CHAR(3) DEFAULT 'INR'
);

SELECT * FROM accounts;


--TRANSACTION TABLE
CREATE TABLE transactions (
  transaction_id BIGSERIAL PRIMARY KEY,
  account_id INT REFERENCES accounts(account_id),
  txn_timestamp TIMESTAMP NOT NULL,
  txn_type VARCHAR(30) CHECK (txn_type IN ('deposit','withdrawal','transfer_in','transfer_out','fee','interest')),
  amount NUMERIC(14,2) NOT NULL,
  channel VARCHAR(20),
  description TEXT
);

SELECT * FROM transactions;


CREATE INDEX idx_transactions_account_time ON transactions(account_id, txn_timestamp);
```

```sql

--LOAN TABLE
CREATE TABLE loans (
  loan_id SERIAL PRIMARY KEY,
  account_id INT REFERENCES accounts(account_id),
  loan_type VARCHAR(30) CHECK (loan_type IN ('home','auto','personal','education','gold','business')),
  principal NUMERIC(14,2) NOT NULL,
  interest_rate NUMERIC(5,2) NOT NULL,
  term_months INT NOT NULL,
  start_date DATE NOT NULL,
  status VARCHAR(20) CHECK (status IN ('active','closed','defaulted'))
);

SELECT * FROM loans;



--MERCHANTS TABLE
CREATE TABLE merchants (
  merchant_id SERIAL PRIMARY KEY,
  merchant_name VARCHAR(120) NOT NULL,
  category VARCHAR(40),
  city VARCHAR(60),
  state VARCHAR(60)
);

SELECT * FROM merchants;



--CARDS TABLE
CREATE TABLE cards (
  card_id SERIAL PRIMARY KEY,
  account_id INT REFERENCES accounts(account_id),
  card_type VARCHAR(20) CHECK (card_type IN ('debit','credit')),
  network VARCHAR(20) CHECK (network IN ('VISA','RuPay','Mastercard')),
  masked_pan VARCHAR(32),
  issued_on DATE,
  expires_on DATE,
  status VARCHAR(20) CHECK (status IN ('active','blocked','expired')),
  credit_limit NUMERIC(14,2)
);

SELECT * FROM cards;


--CARD TRANSACTION TABLE
CREATE TABLE card_transactions (
  card_txn_id BIGSERIAL PRIMARY KEY,
  card_id INT REFERENCES cards(card_id),
  merchant_id INT REFERENCES merchants(merchant_id),
  txn_timestamp TIMESTAMP NOT NULL,
  txn_type VARCHAR(20) CHECK (txn_type IN ('purchase','refund','cash_advance')),
  amount NUMERIC(14,2) NOT NULL
);

SELECT * FROM card_transactions;


SELECT * FROM branches;
SELECT * FROM customers;
SELECT * FROM accounts ;
SELECT * FROM transactions;
SELECT * FROM loans;
```

```sql
SELECT * FROM merchants;
SELECT * FROM cards;
SELECT * FROM card_transactions;


-- ================================
-- 1) CUSTOMER / ACCOUNT INSIGHTS
-- ================================

-- 1. Customers by state (distribution)
SELECT state,COUNT(customer_id) AS Total_customers
FROM customers
GROUP BY state
ORDER BY Total_customers DESC;

-- 2. New customers by month
SELECT DATE_TRUNC('month', created_at) AS month, COUNT(customer_id) AS new_customers
FROM customers
GROUP BY 1
ORDER BY 1;

-- 3. Accounts by status per branch
SELECT b.branch_id,b.branch_name,a.status,COUNT(account_id) AS total_accounts
FROM branches b
LEFT JOIN
accounts a
ON b.branch_id=a.branch_id
GROUP BY b.branch_id,b.branch_name,a.status
ORDER BY b.branch_id,a.status;

-- 4. Top branches by total deposit balance (sum of balances)
SELECT b.branch_id,b.branch_name,SUM(a.balance) AS total_diposit_balance
FROM branches b
LEFT JOIN
accounts a
ON b.branch_id=a.branch_id
GROUP BY b.branch_id,b.branch_name
ORDER BY total_diposit_balance DESC;

-- 5. Top 20 customers by total balance
SELECT c.customer_id,c.full_name,c.city,SUM(a.balance) AS total_balance
FROM customers c
JOIN
accounts a
ON c.customer_id=a.customer_id
GROUP BY c.customer_id,c.full_name,c.city
ORDER BY total_balance DESC
LIMIT 20;

-- 6. Average balance by account type
SELECT account_type,ROUND(AVG(balance),2) AS avg_balance
FROM accounts
GROUP BY account_type;

-- 7. Accounts opened per month (trend)
SELECT DATE_TRUNC('Month',opened_on), COUNT(*) AS accounts_opened
FROM accounts
GROUP BY 1
ORDER BY 1;

-- 8. Product penetration per customer (#accounts, #cards, #loans)
WITH acc AS(
SELECT c.customer_id,COUNT(a.*) AS n_accounts
FROM customers c
```

```sql
JOIN accounts a
ON c.customer_id=a.customer_id
GROUP BY c.customer_id
),
card AS(
SELECT a.customer_id,COUNT(c.*) AS n_cards
FROM accounts a
JOIN cards c
ON a.account_id=c.account_id
GROUP BY a.customer_id
),
loans AS(
SELECT a.customer_id,COUNT(l.*) AS n_loans
FROM accounts a
JOIN loans l
ON a.account_id=l.account_id
GROUP BY a.customer_id
)
SELECT c.customer_id,c.full_name,
COALESCE(acc.n_accounts,0) AS accounts,
COALESCE(card.n_cards,0) AS cards,
COALESCE(loans.n_loans,0) AS loans
FROM customers c
LEFT JOIN acc ON acc.customer_id=c.customer_id
LEFT JOIN card ON card.customer_id=c.customer_id
LEFT JOIN loans ON loans.customer_id=c.customer_id
ORDER BY c.customer_id,
c.customer_id;

-- 9. Customers with no active accounts (anti-join)
SELECT c.customer_id,c.full_name,a.status
FROM customers c
LEFT JOIN accounts a
ON c.customer_id=a.customer_id
WHERE a.status<>'active'
GROUP BY c.customer_id,c.full_name,a.status
ORDER BY c.customer_id;


-- ============================
-- 2) TRANSACTION ANALYTICS
-- ============================

-- 10. Channel mix in the last 90 days
SELECT channel, COUNT(*) AS txn_count, ROUND(SUM(amount),2) AS total_amount
FROM transactions
WHERE txn_timestamp >= CURRENT_DATE - INTERVAL '90 days'
GROUP BY channel
ORDER BY txn_count DESC;

-- 11. Customers payments trend via different methods
SELECT a.account_id,c.full_name,t.channel,SUM(t.amount)
FROM accounts a
JOIN
transactions t
ON a.account_id=t.account_id
JOIN
customers c
ON a.customer_id=c.customer_id
WHERE t.description IN ('Withdrawal','Transfer Out','Interest','Fee')
GROUP BY a.account_id,c.full_name,t.channel
ORDER BY a.account_id;

-- 12. Monthly transaction counts
```

```sql
SELECT EXTRACT(YEAR FROM txn_timestamp ) AS t_year,TO_CHAR(txn_timestamp,'Month') AS t_month,COUNT(*) AS total_transactions
FROM transactions
GROUP BY t_year,t_month
ORDER BY EXTRACT(YEAR FROM txn_timestamp);

-- 13. Average transaction amount by type
SELECT * FROM transactions;
SELECT txn_type,ROUND(AVG(amount),2) AS avg_transaction
FROM transactions
GROUP BY txn_type;

-- 14. Channel mix in last 90 days
SELECT channel, COUNT(*) AS transaction_count
FROM transactions
WHERE txn_timestamp>= CURRENT_DATE - INTERVAL '90 days'
GROUP BY channel
ORDER BY transaction_count DESC;

-- 15. Weekend transactions (Sat/Sun)
SELECT TO_CHAR(txn_timestamp, 'Day') AS txn_day,COUNT(*) AS txn_count
FROM transactions
WHERE EXTRACT (DOW FROM txn_timestamp) IN (0,6)
GROUP BY txn_day;

---- 16. Net inflow per account in last 90 days
WITH acc AS (
  SELECT t.account_id, COUNT(*) AS n_txn
  FROM transactions t
  GROUP BY t.account_id
),
txn AS (
  SELECT t.account_id, SUM(t.amount) AS net_inflow_90d
  FROM transactions t
  WHERE t.txn_timestamp >= CURRENT_DATE - INTERVAL '90 days'
  GROUP BY t.account_id
)
SELECT a.account_id,
       c.full_name,
       COALESCE(acc.n_txn, 0)          AS total_transaction,
       COALESCE(txn.net_inflow_90d, 0) AS total_netflow
FROM accounts a
JOIN customers c ON c.customer_id = a.customer_id
LEFT JOIN acc ON acc.account_id = a.account_id
LEFT JOIN txn ON txn.account_id = a.account_id
ORDER BY a.account_id,
         c.customer_id;

-- 17. Accounts with no transactions ever
SELECT c.customer_id,c.full_name,t.account_id,t.transaction_id
FROM customers c
JOIN accounts a ON c.customer_id=a.customer_id
JOIN transactions t ON a.account_id=t.account_id
WHERE t.transaction_id IS NULL
ORDER BY c.customer_id;

-- 18. Transactions per branch last month
SELECT b.branch_id,b.branch_name,COUNT(*) AS txn_count
FROM branches b
JOIN accounts a ON b.branch_id=a.branch_id
JOIN transactions t ON a.account_id=t.account_id
WHERE txn_timestamp>= DATE_TRUNC('Month',CURRENT_DATE) - INTERVAL '1 month'
AND txn_timestamp < DATE_TRUNC('Month',CURRENT_DATE)
GROUP BY b.branch_id,b.branch_name
```

```sql
ORDER BY b.branch_id;

-- 19. Days since last transaction per account (snapshot)
SELECT account_id,AGE(CURRENT_DATE,MAX(txn_timestamp)::DATE) AS days_since_last_transaction
FROM transactions
GROUP BY 1
ORDER BY 1;

-- 20. Outflow vs inflow by channel by branch (last 60 days)
SELECT b.branch_id,b.branch_name,
       ROUND(SUM(CASE WHEN t.txn_type IN ('withdrawal','transfer_out','fee') THEN -t.amount ELSE 0 END),
2) AS outflows,
       ROUND(SUM(CASE WHEN t.txn_type IN ('deposit','transfer_in','interest') THEN t.amount ELSE 0 END),
2)  AS inflows
FROM branches b
JOIN accounts a ON b.branch_id=a.branch_id
JOIN transactions t ON a.account_id=t.account_id
WHERE t.txn_timestamp >= CURRENT_DATE - INTERVAL '60 days'
GROUP BY b.branch_id,b.branch_name
ORDER BY  b.branch_id;


-- ==================
-- 3) LOAN ANALYTICS
-- ==================

-- 21. Loan count by type
SELECT loan_type,COUNT(*) AS ln_count
FROM loans
GROUP BY 1;

-- 22. Average interest rate by loan type
SELECT loan_type,ROUND(AVG(interest_rate),2) AS avg_intersest
FROM loans
GROUP BY 1;

-- 23. Loans started per quarter
SELECT DATE_TRUNC('quarter',start_date) AS str_qtr,COUNT(*) AS ln_count
FROM loans
GROUP BY 1
ORDER BY 1;

-- 24. EMIs per loan(simple interest)
SELECT * , ROUND((principal * interest_rate * (term_months/12)/100),2) AS EMI
FROM loans
ORDER BY loan_id;

-- 25. Total EMI & loans runs per customer
SELECT c.customer_id,c.full_name,COUNT(l.*) AS total_loans,SUM(ROUND((l.principal * l.interest_rate * (l.
term_months/12)/100),2)) AS total_EMI
FROM accounts a
JOIN customers c ON a.customer_id=c.customer_id
JOIN loans l ON a.account_id=l.account_id
GROUP BY c.customer_id,c.full_name
ORDER BY c.customer_id;

-- 26. Customers that hasn't run any loan
SELECT c.customer_id,c.full_name
FROM customers c
JOIN accounts a ON c.customer_id=a.customer_id
LEFT JOIN loans l ON a.account_id=l.account_id
GROUP BY c.customer_id,c.full_name
HAVING COUNT(l.*)=0
ORDER BY c.customer_id;
```

```sql

-- 27. Top customers by total principal (across their loans)
SELECT c.customer_id,c.full_name,
SUM(l.principal) AS loan_principal
FROM customers c
JOIN accounts a ON c.customer_id=a.customer_id
JOIN loans l ON a.account_id=l.loan_id
GROUP BY c.customer_id,c.full_name
ORDER BY loan_principal DESC
LIMIT 50;

-- 28. Default rate by loan type
SELECT loan_type, ROUND(AVG(interest_rate),2) AS default_rate
FROM loans
WHERE status IN('defaulted')
GROUP BY loan_type;

-- 29. loan complition status
SELECT status, COUNT(*) AS no_of_customers
FROM loans
GROUP BY status
HAVING status='closed';


-- ==========================
-- 4) CARD & MERCHANT INSIGHT
-- ==========================

-- 30. Cards by network and status
SELECT network,status,COUNT(*) AS n_cards
FROM cards
GROUP BY network,status
ORDER BY network,status;

-- 31. Average credit limit by network
SELECT network,ROUND(AVG(credit_limit),2) AS avg_credit_limit
FROM cards
GROUP BY 1
ORDER BY 1;

-- 32. Purchases last month by merchant category
SELECT m.category,SUM(c.amount) AS total_purchase_amount
FROM merchants m
JOIN card_transactions c ON m.merchant_id=c.merchant_id
WHERE c.txn_timestamp>=DATE_TRUNC('Month',CURRENT_DATE) - INTERVAL '1 month'
AND c.txn_timestamp<DATE_TRUNC('Month',CURRENT_DATE)
AND c.txn_type='purchase'
GROUP BY m.category
ORDER BY total_purchase_amount DESC;

-- 33.TOP 50 customer by card uses when purchasing(All time)
SELECT c.customer_id,c.full_name,
COUNT(CASE WHEN ct.txn_type='purchase' THEN ct.card_txn_id ELSE 0 END) AS n_transactions,
SUM(CASE WHEN ct.txn_type='purchase' THEN ct.amount ELSE 0 END) AS total_spent
FROM customers c
JOIN accounts a ON c.customer_id = a.customer_id
JOIN cards ca ON a.account_id = ca.account_id
JOIN card_transactions ct ON ca.card_id = ct.card_id
GROUP BY c.customer_id,c.full_name
ORDER BY total_spent DESC
LIMIT 50;

-- 34.TOP 100 customer by card uses(in this year)
SELECT c.customer_id,c.full_name,
```

```sql
COUNT(CASE WHEN ct.txn_type='purchase' THEN ct.card_txn_id ELSE 0 END) AS n_transactions,
SUM(CASE WHEN ct.txn_type='purchase' THEN ct.amount ELSE 0 END) AS total_spent
FROM customers c
JOIN accounts a ON c.customer_id = a.customer_id
JOIN cards ca ON a.account_id = ca.account_id
JOIN card_transactions ct ON ca.card_id = ct.card_id
WHERE ct.txn_timestamp>= CURRENT_DATE - INTERVAL '1 year'
AND ct.txn_timestamp<'2025-01-01'
GROUP BY c.customer_id,c.full_name
ORDER BY total_spent DESC
LIMIT 100;

-- 35. Cash advance total per card (last 90 days)
SELECT card_id,
SUM(CASE WHEN txn_type='cash_advance' THEN amount ELSE 0 END) AS total_spent
FROM card_transactions
WHERE txn_timestamp>= CURRENT_DATE - INTERVAL '90 days'
GROUP BY card_id
ORDER BY total_spent DESC;

-- 36. Top 30 merchants by spend in last 60 days
SELECT m.merchant_id,m.merchant_name,m.category,
SUM(ct.amount) AS total_spent
FROM merchants m
JOIN card_transactions ct ON m.merchant_id = ct.merchant_id
WHERE ct.txn_timestamp>= CURRENT_DATE - INTERVAL '60 days'
GROUP BY m.merchant_id,m.merchant_name,m.category
ORDER BY total_spent DESC
LIMIT 30;



-- 37. Customers with cards but no card transactions
SELECT c.customer_id,c.full_name,COUNT(ct.*) AS n_transactions
FROM customers c
JOIN accounts a ON c.customer_id = a.customer_id
JOIN cards ca ON a.account_id = ca.account_id
LEFT JOIN card_transactions ct ON ca.card_id = ct.card_id
GROUP BY c.customer_id,c.full_name
HAVING COUNT(ct.*) IS NULL
ORDER BY c.customer_id,c.full_name;


-- ======================
-- 5) CUSTOMER 360 VIEWS
-- ======================

-- 38. Cardholders per customer
SELECT c.customer_id,c.full_name,COUNT(ca.*) AS n_cards
FROM customers c
LEFT JOIN accounts a ON c.customer_id = a.customer_id
LEFT JOIN cards ca ON a.account_id = ca.account_id
GROUP BY c.customer_id,c.full_name
ORDER BY c.customer_id,c.full_name;

-- 39. Total card spend per customer (lifetime)
SELECT c.customer_id,c.full_name,SUM(ct.amount) AS n_cards
FROM customers c
LEFT JOIN accounts a ON c.customer_id = a.customer_id
LEFT JOIN cards ca ON a.account_id = ca.account_id
LEFT JOIN card_transactions ct ON ca.card_id = ct.card_id
GROUP BY c.customer_id,c.full_name
HAVING SUM(ct.amount) IS NOT NULL
```

```sql
ORDER BY c.customer_id,c.full_name;

-- 40. Customers with any loan and at least one card
SELECT c.customer_id,c.full_name,
COUNT(l.*) AS n_loans,
COUNT(ca.*) AS n_cards
FROM customers c
LEFT JOIN accounts a ON c.customer_id = a.customer_id
LEFT JOIN loans l ON a.account_id = l.account_id
LEFT JOIN cards ca ON a.account_id = ca.account_id
GROUP BY c.customer_id,c.full_name
HAVING COUNT(ca.*) >= 1 AND COUNT(l.*) >=1
ORDER BY c.customer_id,c.full_name;

-- 41. Salary accounts by branch
SELECT b.branch_id,b.branch_name,
(COUNT(a.*) FILTER(WHERE a.account_type = 'salary')) AS salary_accounts
FROM branches b
LEFT JOIN accounts a
ON b.branch_id = a.branch_id
GROUP BY b.branch_id,b.branch_name
ORDER BY b.branch_id,b.branch_name;


-- ============================
-- 6) PERIOD & KPI SNAPSHOTS
-- ============================

-- 42. Interest credited YTD
SELECT (SUM(amount) FILTER(WHERE txn_type = 'interest')) AS YTD_interest
FROM transactions
WHERE txn_timestamp>=DATE_TRUNC('Year',CURRENT_DATE);

-- 43. Accounts opened by year
SELECT EXTRACT(YEAR FROM opened_on) AS years,COUNT(*) n_accounts
FROM accounts
GROUP BY 1
ORDER BY 1;

-- 44. Total transactions amounts by year
SELECT EXTRACT(YEAR FROM txn_timestamp) AS years,
txn_type,
SUM(CASE WHEN txn_type IN ('fee','transfer_out','withdrawal') THEN -amount
   ELSE amount END
   )AS total_amount
FROM transactions
GROUP BY years,txn_type
ORDER BY years,txn_type;

-- 45. Customers opened accounts in this year
SELECT c.customer_id,c.full_name,
COUNT(a.*) n_accounts
FROM customers c
JOIN accounts a ON c.customer_id = a.customer_id
WHERE opened_on >= DATE_TRUNC('Year',CURRENT_DATE)
GROUP BY c.customer_id,c.full_name
ORDER BY c.customer_id,c.full_name;

-- 46. Total accounts opened YTD
SELECT SUM(n_accounts) AS total_YTD_accounts
FROM(
SELECT COUNT(a.account_id) AS n_accounts
FROM customers c
JOIN accounts a ON c.customer_id = a.customer_id
```

```sql
WHERE opened_on >= DATE_TRUNC('Year',CURRENT_DATE)
);

-- 47. Transactions completed by cards in this year
SELECT COUNT(*) AS YTD_transactions
FROM card_transactions
WHERE txn_timestamp>= DATE_TRUNC('Year',CURRENT_DATE);

-- 48. Customers that buys card in this year
SELECT c.customer_id,c.full_name,COUNT(ca.*) AS n_cards
FROM customers c
JOIN accounts a ON c.customer_id = a.customer_id
JOIN cards ca ON a.account_id = ca.account_id
WHERE ca.issued_on >= DATE_TRUNC('Year',CURRENT_DATE)
GROUP BY c.customer_id,c.full_name
ORDER BY c.customer_id,c.full_name;

-- 49. Cards expiring in next 60 days
SELECT card_id,account_id,network,expires_on
FROM cards
WHERE expires_on > CURRENT_DATE AND expires_on <= CURRENT_DATE + INTERVAL '60 days'
ORDER BY expires_on;


-- ============================
-- 7) CUSTOMERS BALANCE SUMMARY
-- ============================

-- 50. Cureent balance left in accounts by customers(who have accounts or cards or both)
WITH
acct AS (
  SELECT customer_id, SUM(balance) AS sum_account_balance
  FROM accounts
  GROUP BY customer_id
),
txns AS (
  SELECT a.customer_id, SUM(t.amount) AS sum_txn_amount
  FROM transactions t
  JOIN accounts a ON t.account_id = a.account_id
  GROUP BY a.customer_id
),
card_txns AS (
  SELECT a.customer_id, SUM(ct.amount) AS sum_card_txn
  FROM cards ca
  JOIN accounts a ON ca.account_id = a.account_id
  JOIN card_transactions ct ON ca.card_id = ct.card_id
  GROUP BY a.customer_id
)
SELECT
  c.customer_id,
  c.full_name,
  COALESCE(ac.sum_account_balance,0) AS total_account_balance,
  COALESCE(tx.sum_txn_amount,0)      AS total_transactions,
  COALESCE(cc.sum_card_txn,0)       AS total_card_transactions,
  ROUND(
    COALESCE(ac.sum_account_balance,0)
    + COALESCE(tx.sum_txn_amount,0)
    - COALESCE(cc.sum_card_txn,0)
  ,2) AS net_balance
FROM customers c
LEFT JOIN acct ac ON c.customer_id = ac.customer_id
LEFT JOIN txns tx ON c.customer_id = tx.customer_id
LEFT JOIN card_txns cc ON c.customer_id = cc.customer_id
-- optional: show only customers who have some accounts/cards/transactions
```

```sql
WHERE COALESCE(ac.sum_account_balance,0) <> 0
   OR COALESCE(tx.sum_txn_amount,0) <> 0
   OR COALESCE(cc.sum_card_txn,0) <> 0
ORDER BY c.customer_id;

-- 51. Cureent balance left in customers account id(who have accounts or cards or both)
SELECT
  c.customer_id,
  c.full_name,
  a.account_id,
  a.balance AS account_balance,
  ROUND(
    a.balance
    + COALESCE(t.txn_sum,0)      -- transactions on this account
    - COALESCE(ct.card_txn_sum,0) -- card transactions on cards linked to this account
  , 2) AS net_balance
FROM customers c
JOIN accounts a ON c.customer_id = a.customer_id
LEFT JOIN (
  SELECT account_id, SUM(amount) AS txn_sum
  FROM transactions
  GROUP BY account_id
) t ON a.account_id = t.account_id
LEFT JOIN (
  SELECT ca.account_id, SUM(ct.amount) AS card_txn_sum
  FROM cards ca
  JOIN card_transactions ct ON ca.card_id = ct.card_id
  GROUP BY ca.account_id
) ct ON a.account_id = ct.account_id
ORDER BY c.customer_id,
  c.full_name;

-- 52. TOP 50 richest customers in our bank
WITH
acct AS (
  SELECT customer_id, SUM(balance) AS sum_account_balance
  FROM accounts
  GROUP BY customer_id
),
txns AS (
  SELECT a.customer_id, SUM(t.amount) AS sum_txn_amount
  FROM transactions t
  JOIN accounts a ON t.account_id = a.account_id
  GROUP BY a.customer_id
),
card_txns AS (
  SELECT a.customer_id, SUM(ct.amount) AS sum_card_txn
  FROM cards ca
  JOIN accounts a ON ca.account_id = a.account_id
  JOIN card_transactions ct ON ca.card_id = ct.card_id
  GROUP BY a.customer_id
)
SELECT
  c.customer_id,
  c.full_name,
  COALESCE(ac.sum_account_balance,0) AS total_account_balance,
  COALESCE(tx.sum_txn_amount,0)      AS total_transactions,
  COALESCE(cc.sum_card_txn,0)       AS total_card_transactions,
  ROUND(
    COALESCE(ac.sum_account_balance,0)
    + COALESCE(tx.sum_txn_amount,0)
    - COALESCE(cc.sum_card_txn,0)
  ,2) AS net_balance
FROM customers c
```

```sql
LEFT JOIN acct ac ON c.customer_id = ac.customer_id
LEFT JOIN txns tx ON c.customer_id = tx.customer_id
LEFT JOIN card_txns cc ON c.customer_id = cc.customer_id
WHERE COALESCE(ac.sum_account_balance,0) <> 0
ORDER BY net_balance DESC
LIMIT 50;

-- 53. Accounts that have low balance(Below 10000/-)
SELECT
  c.customer_id,
  c.full_name,
  c.phone,
  a.account_id,
  a.balance AS account_balance,
  ROUND(
    a.balance
    + COALESCE(t.txn_sum,0)      -- transactions on this account
    - COALESCE(ct.card_txn_sum,0) -- card transactions on cards linked to this account
  , 2) AS net_balance
FROM customers c
JOIN accounts a ON c.customer_id = a.customer_id
LEFT JOIN (
  SELECT account_id, SUM(amount) AS txn_sum
  FROM transactions
  GROUP BY account_id
) t ON a.account_id = t.account_id
LEFT JOIN (
  SELECT ca.account_id, SUM(ct.amount) AS card_txn_sum
  FROM cards ca
  JOIN card_transactions ct ON ca.card_id = ct.card_id
  GROUP BY ca.account_id
) ct ON a.account_id = ct.account_id
WHERE (ROUND(
    a.balance
    + COALESCE(t.txn_sum,0)
    - COALESCE(ct.card_txn_sum,0)
  , 2)) <=10000
ORDER BY net_balance;


SELECT COUNT(account_id)
FROM accounts;
```

yono SBI

Filters

- Overview
- Branch Info.
- Customers Info.
- Cards
- Loans
- Customers

Total Customers
750

Total Accounts
1100

Total loans
400

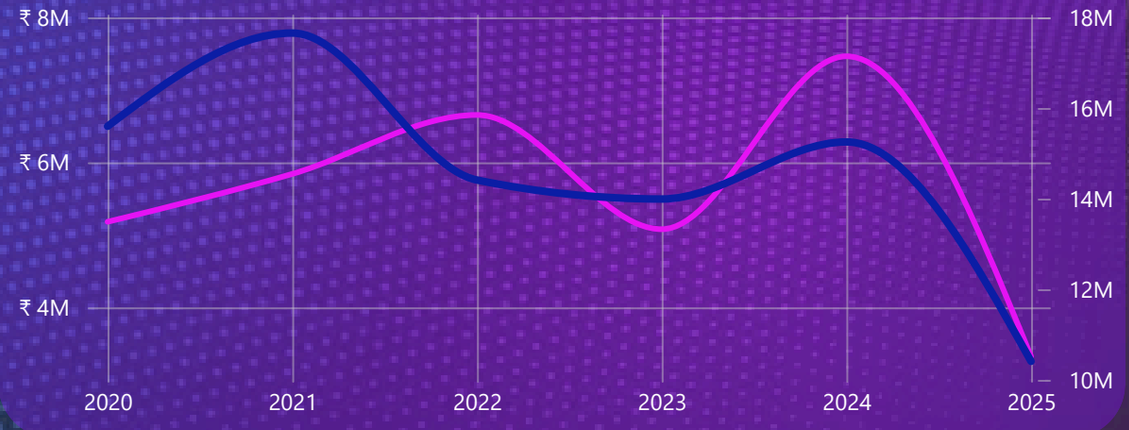
Total Cards
800

Treasury Value

Current Treasury
₹ 3,35,40,279.05

Total Balance Summary
₹ 8,73,11,948.31

Day
Month
Year

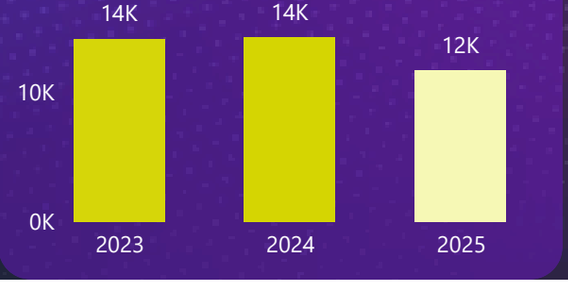


Total loan Amount
₹21,83,76,986.48
85

Total Card Transacted Amount
₹5,50,80,423.89
23K

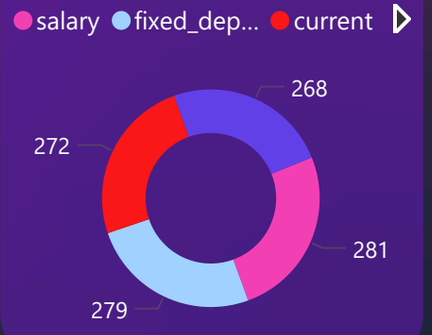
Bank charges
₹ 5,79,67,784…
8068

Transactions made by Year



Card Transactions made by Year

Accounts by Type



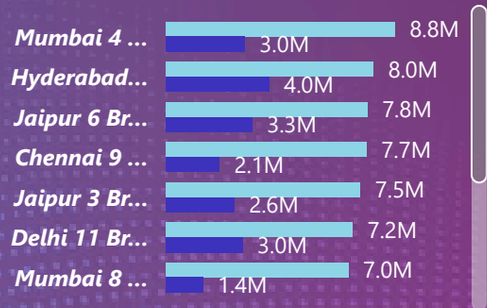
Top 5 Customers

Top 5 Loan Holders

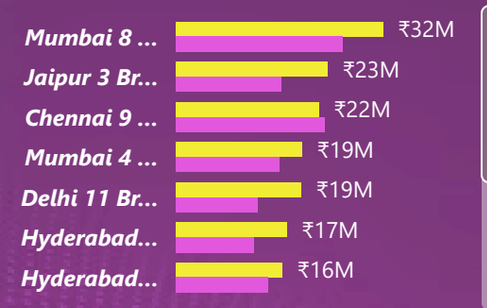
yono SBI

- Overview
- **Branch Info.**
- Customers Info.
- Cards
- Loans
- Customers

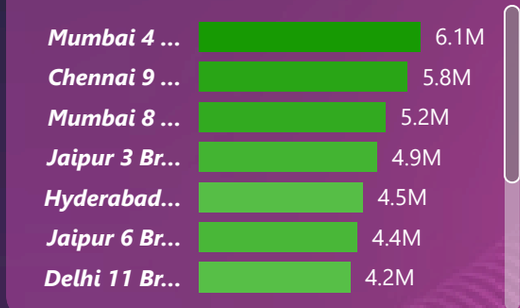
Account Balance



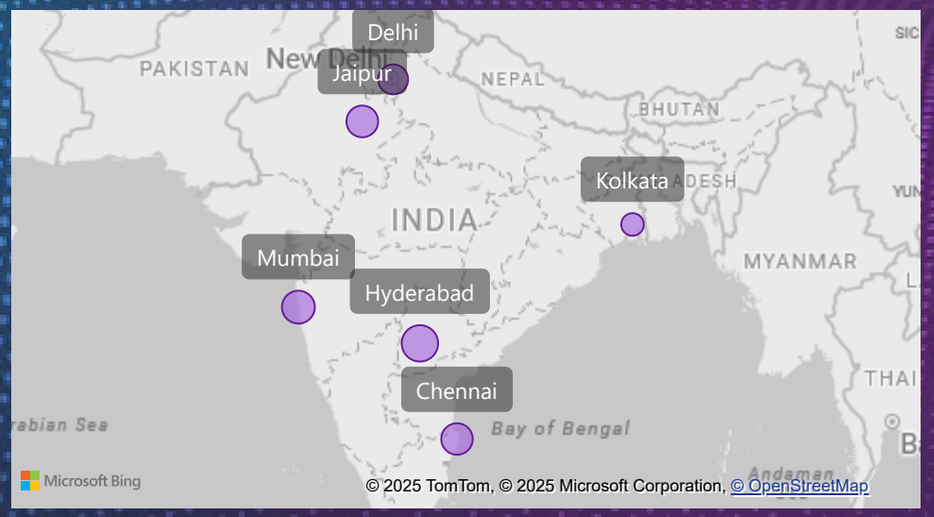
Loans

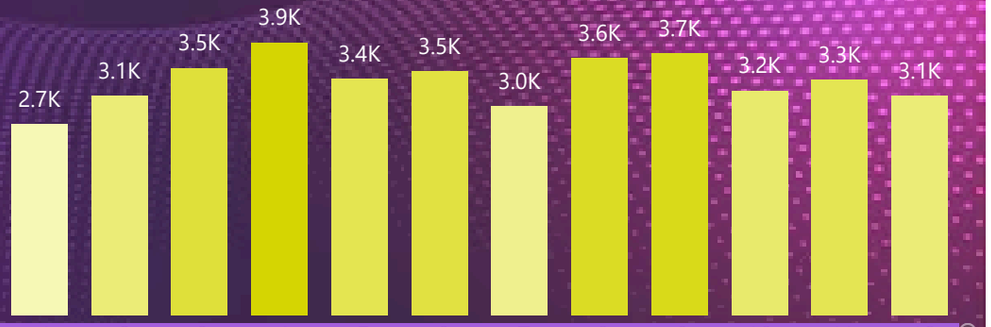


Card Transactions



Public Accounts





Branch Id	Branch Name	No of cards	No of loans	Closed loans
1	Delhi 1 Branch	57	28	7
2	Hyderabad 2 Branch	55	32	6
3	Jaipur 3 Branch	71	38	7
4	Mumbai 4 Branch	89	33	6
5	Hyderabad 5 Branch	64	24	2
6	Jaipur 6 Branch	63	28	9

Day / Hour	1	2	3	4	5	6	7	8	9	10	11	12
10-12 AM	371	396	458	474	420	433	359	463	441	399	424	395
10-12 PM	210	268	283	301	293	308	239	305	304	275	265	281
1-3 AM	438	503	574	641	569	582	521	567	648	546	563	533
1-3 PM	314	396	475	544	416	429	425	466	453	418	412	370
4-6 AM	360	406	422	464	422	378	360	452	430	382	437	371
4-6 PM	355	417	399	477	426	451	338	465	475	385	417	361
7-9 AM	345	358	441	484	385	426	362	453	484	399	394	396

yono SBI

- Overview
- Branch Info.
- **Customers Info.**
- Cards
- Loans
- Customers

49

Name
Aadhya Banerjee

D.O.B
01-28-1955

Email
aadhya.banerjee220@testmail.com

Ph No.
9002007565

City
Ahmedabad

KYC Status
pending

Account Created at
03-01-2020

customer_id: 49

account_id: All | card_id: All | loan_id: All

No. of accounts	No. of Cards	No. of Loans
5	2	3

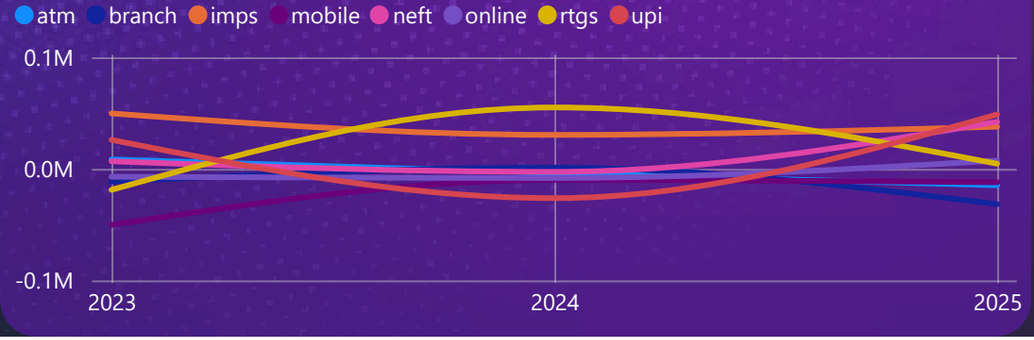
Current Balance
Acc. ID 214 fixed_deposit
₹3,65,348.82
₹3,63,685.43

Account transactions
Total transactions 195
₹1,21,391.56
28-12-2020

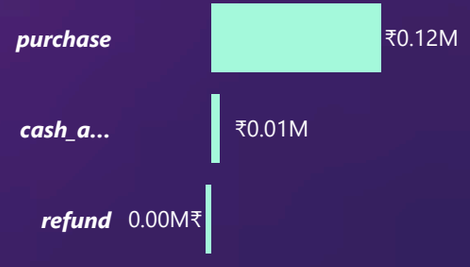
Card Info active
Card ID 380 credit
₹ 1,19,728.17
XXXX-XXXX-XXXX-2092
Mastercard 08-14-2027

Loan Info active
Loan ID 65 gold
₹15,81,931.61
Months ₹ 5,04,971.97
96 30-03-2022

Transactions Summary



Card transactions



Bank charges ₹ 3,24,974.05

Account Balance

yono SBI

Filters

Overview

Branch Info.

Customers Info.

Cards

Loans

Customers

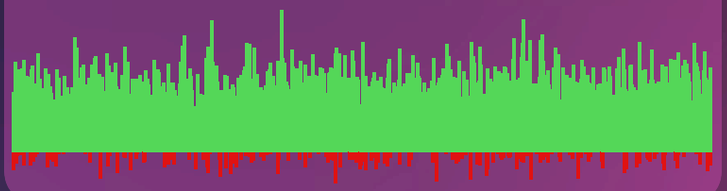
No of Card holders

585

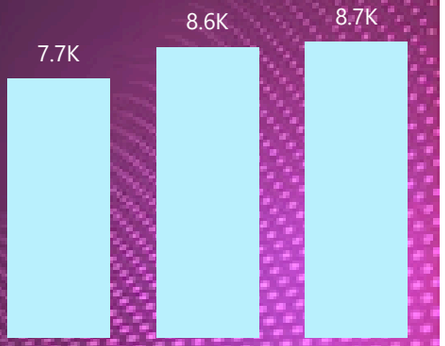
Total Cards

800

Transactions Analysis

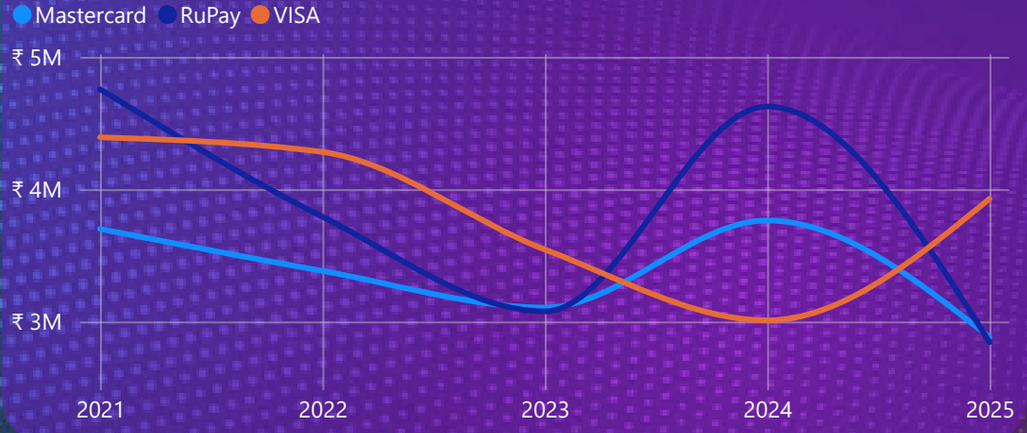


Transaction Timing Trend

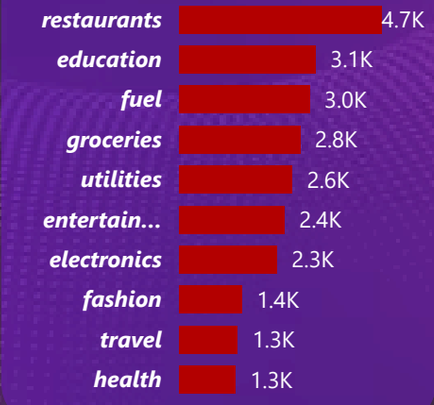


Day Hour	Mastercard	RuPay	VISA
10-12 AM	960	1083	1066
10-12 PM	648	710	738
1-3 AM	1272	1488	1520
1-3 PM	951	1104	1115
4-6 AM	945	1042	1061
4-6 PM	955	1072	1126
7-9 AM	951	1063	1084
7-9 PM	981	1031	1034

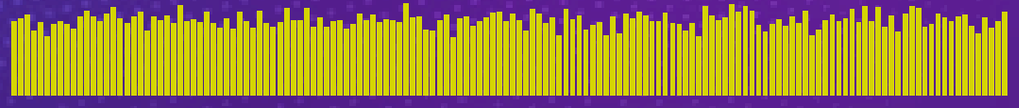
Card transaction by Network



Card transactions by Category

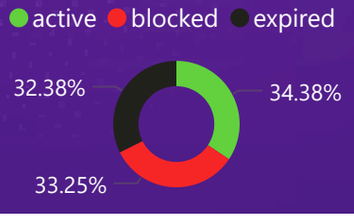


Transaction amount by Merchants



Card type

Card status



Card Summary

Card ID	Type	Network	Card no.	Card limit	Transactions
566	debit	RuPay	XXXX-XXXX-XXXX-9759	₹ 1,48,170.56	₹ 1,46,058.29
680	credit	VISA	XXXX-XXXX-XXXX-4810	₹ 94,606.32	₹ 1,29,474.52
568	debit	RuPay	XXXX-XXXX-XXXX-8904	₹ 1,09,618.01	₹ 1,18,041.05
2	debit	RuPay	XXXX-XXXX-XXXX-4125	₹ 0.00	₹ 1,09,125.17
391	credit	Mastercard	XXXX-XXXX-XXXX-4465	₹ 1,05,851.35	₹ 1,08,816.98
541	credit	RuPay	XXXX-XXXX-XXXX-1324	₹ 1,51,937.56	₹ 1,08,616.13
284	credit	RuPay	XXXX-XXXX-XXXX-6095	₹ 1,01,542.81	₹ 1,07,114.67
773	debit	VISA	XXXX-XXXX-XXXX-8205	₹ 1,25,357.26	₹ 1,06,981.42
349	debit	VISA	XXXX-XXXX-XXXX-8286	₹ 2,49,062.65	₹ 1,06,972.56
386	credit	Mastercard	XXXX-XXXX-XXXX-6986	₹ 2,24,607.27	₹ 1,05,903.97

yono SBI

Filters

- Overview
- Branch Info.
- Customers Info.
- Cards
- **Loans**
- Customers

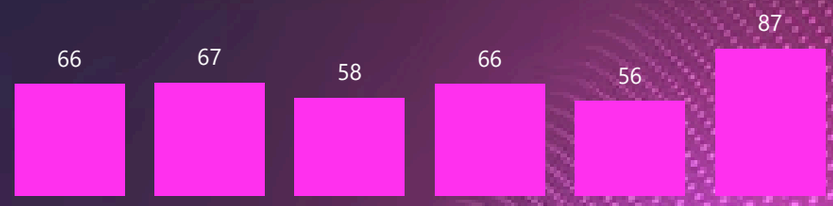
No of Loan holders

331

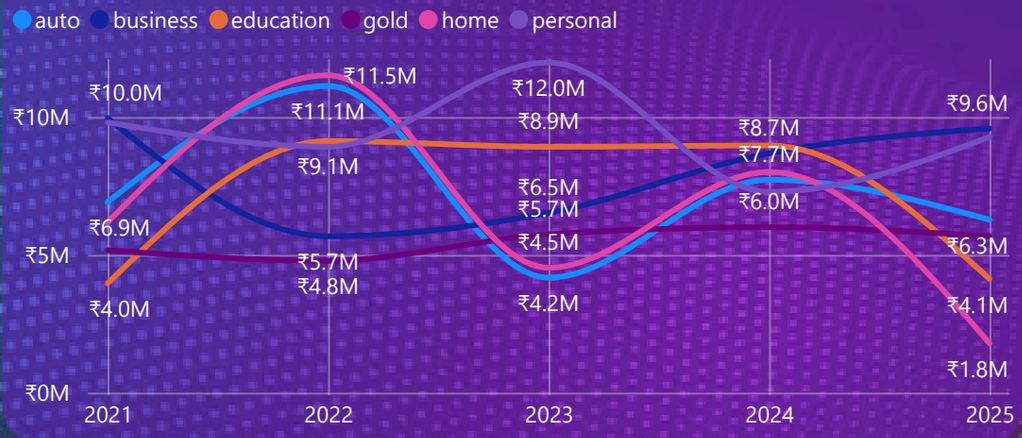
Total Loans

400

Loan Timing Trend



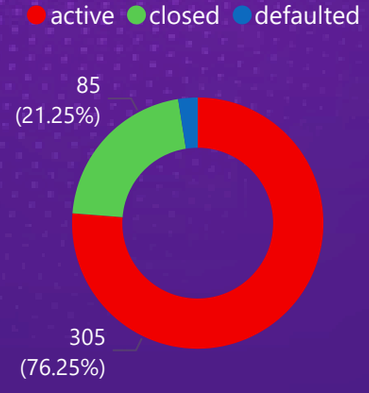
Loan Amount by Loan Type



Month name	auto	business	education	gold	home	personal
Jan	11	7	2	4	6	12
Feb	6	4	6	4	3	7
Mar	4	6	5	5	10	3
Apr	3	7	7	4	6	10
May	7	8	4	4	3	5
Jun	4	4	3	5	6	8
Jul	8	5	7	10	6	7
Aug	5	9	7	8	1	12
Sep	6	6	4	7	4	8
Oct	2	5	3	6	4	6
Nov	3	1	4	2	5	5
Dec	7	5	6	7	2	4

Loan Type

Loan status



Loan Summary

loan id	loan type	Principal	Rate	Interest amount	Tenure	status
326	business	₹14,25,745.82	8.41	₹ 8,39,336.56	84	active
257	auto	₹14,13,480.81	15.83	₹ 44,75,080.24	240	active
275	personal	₹13,33,153.71	11.40	₹ 7,59,897.61	60	active
348	auto	₹13,18,359.74	10.88	₹ 5,73,750.16	48	active
82	education	₹12,83,049.53	12.31	₹ 4,73,830.19	36	active
151	personal	₹12,69,440.97	14.90	₹ 9,45,733.52	60	active
141	personal	₹12,66,699.55	8.17	₹ 5,17,446.77	60	active
318	business	₹12,28,309.42	10.30	₹ 12,65,158.70	120	active
152	home	₹12,25,810.62	16.48	₹ 14,14,095.13	84	active
154	home	₹12,09,147.08	10.65	₹ 1,28,774.16	12	active

yono SBI

Filters

Overview

Branch Info.

Customers Info.

Cards

Loans

Customers

Customer Id	public customers.full_name	Accounts	Cards	Loans	Account Balance	Current Balance	Card txn amount	Loan principal	Loan to Pay
122	Sai Verma	3	0	0	₹3,54,758.49	₹5,47,732.26			
546	Krishna Chakraborty	5	1	3	₹4,98,530.53	₹5,00,345.67	₹ 63,058.22	₹14,17,071.36	₹ 25,70,729.05
503	Navya Ghosh	5	2	1	₹7,42,410.81	₹4,92,346.47	₹ 1,70,406.90	₹4,87,888.76	₹ 12,72,413.89
216	Navya Gupta	3	0	1	₹3,19,611.55	₹4,69,496.04		₹4,33,597.86	₹ 13,51,090.93
330	Aditya Gupta	6	2	5	₹6,12,812.25	₹4,61,976.56	₹ 1,67,286.02	₹23,94,900.14	₹ 68,48,279.37
364	Sai Patel	3	0	0	₹3,04,095.59	₹4,44,259.44			
146	Vivaan Gupta	6	1	1	₹5,37,331.60	₹4,40,393.80	₹ 51,767.96	₹5,34,362.51	₹ 8,56,583.10
52	Aarohi Ghosh	2	0	1	₹4,19,963.10	₹4,17,276.48		₹1,05,539.95	₹ 1,79,344.04
27	Myra Pillai	4	4	1	₹5,32,506.71	₹4,01,067.15	₹ 3,03,009.27	₹4,40,055.25	₹ 10,90,896.96
156	Arjun Mehta	2	0	0	₹3,31,302.99	₹3,78,524.84			
49	Aarohi Mukherjee	5	2	3	₹3,63,685.43	₹3,65,348.82	₹ 1,19,728.17	₹15,81,931.61	₹ 20,86,903.58
157	Vihaan Bose	2	2	3	₹3,59,745.77	₹3,64,147.65	₹ 1,31,808.77	₹4,84,565.54	₹ 11,09,551.55
259	Aarohi Bose	4	2	0	₹2,55,463.00	₹3,57,999.61	₹ 1,24,751.84		
559	Ayaan Iyer	2	0	0	₹2,45,167.34	₹3,52,750.86			
241	Ishaan Nair	4	1	0	₹4,71,600.31	₹3,32,240.90	₹ 87,455.74		
92	Vihaan Mukherjee	2	0	0	₹1,86,017.45	₹3,30,649.56			
449	Vihaan Mehta	2	1	0	₹2,81,995.32	₹3,30,353.24	₹ 66,253.05		
460	Aarohi Mukherjee	3	3	1	₹4,00,336.23	₹3,24,629.38	₹ 1,99,612.65	₹5,57,791.05	₹ 10,43,627.05
494	Ira Nair	2	1	0	₹2,62,973.91	₹3,23,331.67	₹ 68,027.10		
505	Navya Das	3	0	1	₹3,85,053.92	₹3,21,721.94		₹55,814.40	₹ 78,475.05
685	Sara Chakraborty	3	0	0	₹2,50,950.62	₹3,16,801.03			
41	Ayaan Reddy	5	6	1	₹5,73,077.09	₹3,14,434.56	₹ 4,12,604.10	₹7,78,509.35	₹ 10,45,226.65
268	Sanvi Chatterjee	5	2	1	₹5,30,223.73	₹3,11,301.54	₹ 1,40,005.56	₹11,840.50	₹ 13,208.08
598	Navya Singh	2	0	0	₹2,63,568.86	₹3,10,010.83			
486	Sara Mehta	4	2	1	₹3,77,984.57	₹3,00,686.05	₹ 1,47,524.84	₹7,68,604.97	₹ 19,27,661.26
438	Ayaan Reddy	3	1	2	₹2,42,856.63	₹2,93,065.93	₹ 74,739.91	₹13,82,309.60	₹ 34,95,806.41
328	Rohan Mukherjee	5	4	2	₹2,62,279.17	₹2,92,948.36	₹ 2,21,472.08	₹12,50,908.43	₹ 19,56,181.91
98	Sara Kulkarni	2	1	1	₹3,01,667.36	₹2,87,596.74	₹ 62,213.93	₹5,15,409.16	₹ 17,64,760.96
292	Sanvi Mishra	3	1	1	₹2,09,395.71	₹2,81,749.45	₹ 93,235.17	₹3,38,169.92	₹ 4,71,780.86
639	Ayaan Singh	2	0	0	₹1,33,072.77	₹2,79,772.63			
395	Ananya Iyer	3	2	0	₹4,26,354.53	₹2,76,807.50	₹ 1,44,743.20		
589	Arjun Chakraborty	2	1	2	₹2,32,036.48	₹2,76,617.92	₹ 44,362.83	₹6,42,827.97	₹ 7,74,014.70