

SQL Code

```
1 --BRANCHES TABLE
2 CREATE TABLE branches (
3     branch_id SERIAL PRIMARY KEY,
4     branch_name VARCHAR(100) NOT NULL,
5     ifsc_code VARCHAR(20) UNIQUE NOT NULL,
6     city VARCHAR(60),
7     state VARCHAR(60),
8     manager_name VARCHAR(100),
9     opened_on DATE
10 );
11
12 SELECT * FROM branches;
13
14
15 --CUSTOMER TABLE
16 CREATE TABLE customers (
17     customer_id SERIAL PRIMARY KEY,
18     full_name VARCHAR(120) NOT NULL,
19     email VARCHAR(120) UNIQUE,
20     phone VARCHAR(20),
21     city VARCHAR(60),
22     state VARCHAR(60),
23     dob DATE,
24     kyc_status VARCHAR(20) CHECK (kyc_status IN ('verified','pending','rejected')),
25     created_at DATE DEFAULT CURRENT_DATE
26 );
27
28 SELECT * FROM customers;
29
30
31 --ACCOUNTS TABLE
32 CREATE TABLE accounts (
33     account_id SERIAL PRIMARY KEY,
34     customer_id INT REFERENCES customers(customer_id),
35     branch_id INT REFERENCES branches(branch_id),
36     account_type VARCHAR(30) CHECK (account_type IN ('savings','current','salary','fixed_deposit')),
37     opened_on DATE,
38     status VARCHAR(20) CHECK (status IN ('active','dormant','closed')),
39     balance NUMERIC(14,2) DEFAULT 0,
40     currency CHAR(3) DEFAULT 'INR'
41 );
42
43 SELECT * FROM accounts;
44
45
46 --TRANSACTION TABLE
47 CREATE TABLE transactions (
48     transaction_id BIGSERIAL PRIMARY KEY,
49     account_id INT REFERENCES accounts(account_id),
50     txn_timestamp TIMESTAMP NOT NULL,
51     txn_type VARCHAR(30) CHECK (txn_type IN ('deposit','withdrawal','transfer_in','transfer_out','fee','interest')),
52     amount NUMERIC(14,2) NOT NULL,
53     channel VARCHAR(20),
54     description TEXT
55 );
56
57 SELECT * FROM transactions;
58
59
60 CREATE INDEX idx_transactions_account_time ON transactions(account_id, txn_timestamp);
```

```

61
62 --LOAN TABLE
63 CREATE TABLE loans (
64     loan_id SERIAL PRIMARY KEY,
65     account_id INT REFERENCES accounts(account_id),
66     loan_type VARCHAR(30) CHECK (loan_type IN ('home','auto','personal','education','gold','business')),
67     principal NUMERIC(14,2) NOT NULL,
68     interest_rate NUMERIC(5,2) NOT NULL,
69     term_months INT NOT NULL,
70     start_date DATE NOT NULL,
71     status VARCHAR(20) CHECK (status IN ('active','closed','defaulted'))
72 );
73
74 SELECT * FROM loans;
75
76
77
78 --MERCHANTS TABLE
79 CREATE TABLE merchants (
80     merchant_id SERIAL PRIMARY KEY,
81     merchant_name VARCHAR(120) NOT NULL,
82     category VARCHAR(40),
83     city VARCHAR(60),
84     state VARCHAR(60)
85 );
86
87 SELECT * FROM merchants;
88
89
90
91 --CARDS TABLE
92 CREATE TABLE cards (
93     card_id SERIAL PRIMARY KEY,
94     account_id INT REFERENCES accounts(account_id),
95     card_type VARCHAR(20) CHECK (card_type IN ('debit','credit')),
96     network VARCHAR(20) CHECK (network IN ('VISA','RuPay','Mastercard')),
97     masked_pan VARCHAR(32),
98     issued_on DATE,
99     expires_on DATE,
100    status VARCHAR(20) CHECK (status IN ('active','blocked','expired')),
101    credit_limit NUMERIC(14,2)
102 );
103
104 SELECT * FROM cards;
105
106
107 --CARD TRANSACTION TABLE
108 CREATE TABLE card_transactions (
109     card_txn_id BIGSERIAL PRIMARY KEY,
110     card_id INT REFERENCES cards(card_id),
111     merchant_id INT REFERENCES merchants(merchant_id),
112     txn_timestamp TIMESTAMP NOT NULL,
113     txn_type VARCHAR(20) CHECK (txn_type IN ('purchase','refund','cash_advance')),
114     amount NUMERIC(14,2) NOT NULL
115 );
116
117 SELECT * FROM card_transactions;
118
119
120 SELECT * FROM branches;
121 SELECT * FROM customers;
122 SELECT * FROM accounts ;
123 SELECT * FROM transactions;
124 SELECT * FROM loans;

```

```

125 SELECT * FROM merchants;
126 SELECT * FROM cards;
127 SELECT * FROM card_transactions;
128
129
130 -- =====
131 -- 1) CUSTOMER / ACCOUNT INSIGHTS
132 -- =====
133
134 -- 1. Customers by state (distribution)
135 SELECT state,COUNT(customer_id) AS Total_customers
136 FROM customers
137 GROUP BY state
138 ORDER BY Total_customers DESC;
139
140 -- 2. New customers by month
141 SELECT DATE_TRUNC('month', created_at) AS month, COUNT(customer_id) AS new_customers
142 FROM customers
143 GROUP BY 1
144 ORDER BY 1;
145
146 -- 3. Accounts by status per branch
147 SELECT b.branch_id,b.branch_name,a.status,COUNT(account_id) AS total_accounts
148 FROM branches b
149 LEFT JOIN
150 accounts a
151 ON b.branch_id=a.branch_id
152 GROUP BY b.branch_id,b.branch_name,a.status
153 ORDER BY b.branch_id,a.status;
154
155 -- 4. Top branches by total deposit balance (sum of balances)
156 SELECT b.branch_id,b.branch_name,SUM(a.balance) AS total_deposit_balance
157 FROM branches b
158 LEFT JOIN
159 accounts a
160 ON b.branch_id=a.branch_id
161 GROUP BY b.branch_id,b.branch_name
162 ORDER BY total_deposit_balance DESC;
163
164 -- 5. Top 20 customers by total balance
165 SELECT c.customer_id,c.full_name,c.city,SUM(a.balance) AS total_balance
166 FROM customers c
167 JOIN
168 accounts a
169 ON c.customer_id=a.customer_id
170 GROUP BY c.customer_id,c.full_name,c.city
171 ORDER BY total_balance DESC
172 LIMIT 20;
173
174 -- 6. Average balance by account type
175 SELECT account_type,ROUND(AVG(balance),2) AS avg_balance
176 FROM accounts
177 GROUP BY account_type;
178
179 -- 7. Accounts opened per month (trend)
180 SELECT DATE_TRUNC('Month',opened_on), COUNT(*) AS accounts_opened
181 FROM accounts
182 GROUP BY 1
183 ORDER BY 1;
184
185 -- 8. Product penetration per customer (#accounts, #cards, #loans)
186 WITH acc AS(
187 SELECT c.customer_id,COUNT(a.*) AS n_accounts
188 FROM customers c

```

```

189 JOIN accounts a
190 ON c.customer_id=a.customer_id
191 GROUP BY c.customer_id
192 ),
193 card AS(
194 SELECT a.customer_id,COUNT(c.*) AS n_cards
195 FROM accounts a
196 JOIN cards c
197 ON a.account_id=c.account_id
198 GROUP BY a.customer_id
199 ),
200 loans AS(
201 SELECT a.customer_id,COUNT(l.*) AS n_loans
202 FROM accounts a
203 JOIN loans l
204 ON a.account_id=l.account_id
205 GROUP BY a.customer_id
206 )
207 SELECT c.customer_id,c.full_name,
208 COALESCE(acc.n_accounts,0) AS accounts,
209 COALESCE(card.n_cards,0) AS cards,
210 COALESCE(loans.n_loans,0) AS loans
211 FROM customers c
212 LEFT JOIN acc ON acc.customer_id=c.customer_id
213 LEFT JOIN card ON card.customer_id=c.customer_id
214 LEFT JOIN loans ON loans.customer_id=c.customer_id
215 ORDER BY c.customer_id,
216 c.customer_id;
217
218 -- 9. Customers with no active accounts (anti-join)
219 SELECT c.customer_id,c.full_name,a.status
220 FROM customers c
221 LEFT JOIN accounts a
222 ON c.customer_id=a.customer_id
223 WHERE a.status<>'active'
224 GROUP BY c.customer_id,c.full_name,a.status
225 ORDER BY c.customer_id;
226
227
228 =====
229 -- 2) TRANSACTION ANALYTICS
230 =====
231
232 -- 10. Channel mix in the last 90 days
233 SELECT channel, COUNT(*) AS txn_count, ROUND(SUM(amount),2) AS total_amount
234 FROM transactions
235 WHERE txn_timestamp >= CURRENT_DATE - INTERVAL '90 days'
236 GROUP BY channel
237 ORDER BY txn_count DESC;
238
239 -- 11. Customers payments trend via different methods
240 SELECT a.account_id,c.full_name,t.channel,SUM(t.amount)
241 FROM accounts a
242 JOIN
243 transactions t
244 ON a.account_id=t.account_id
245 JOIN
246 customers c
247 ON a.customer_id=c.customer_id
248 WHERE t.description IN ('Withdrawal','Transfer Out','Interest','Fee')
249 GROUP BY a.account_id,c.full_name,t.channel
250 ORDER BY a.account_id;
251
252 -- 12. Monthly transaction counts

```

```

253 SELECT EXTRACT(YEAR FROM txn_timestamp) AS t_year, TO_CHAR(txn_timestamp, 'Month') AS t_month, COUNT(*) AS
total_transactions
254   FROM transactions
255   GROUP BY t_year, t_month
256   ORDER BY EXTRACT(YEAR FROM txn_timestamp);
257
258 -- 13. Average transaction amount by type
259 SELECT * FROM transactions;
260 SELECT txn_type, ROUND(AVG(amount), 2) AS avg_transaction
261   FROM transactions
262   GROUP BY txn_type;
263
264 -- 14. Channel mix in last 90 days
265 SELECT channel, COUNT(*) AS transaction_count
266   FROM transactions
267 WHERE txn_timestamp >= CURRENT_DATE - INTERVAL '90 days'
268   GROUP BY channel
269   ORDER BY transaction_count DESC;
270
271 -- 15. Weekend transactions (Sat/Sun)
272 SELECT TO_CHAR(txn_timestamp, 'Day') AS txn_day, COUNT(*) AS txn_count
273   FROM transactions
274 WHERE EXTRACT(DOW FROM txn_timestamp) IN (0, 6)
275   GROUP BY txn_day;
276
277 ---- 16. Net inflow per account in last 90 days
278 WITH acc AS (
279   SELECT t.account_id, COUNT(*) AS n_txn
280     FROM transactions t
281     GROUP BY t.account_id
282 ),
283 txn AS (
284   SELECT t.account_id, SUM(t.amount) AS net_inflow_90d
285     FROM transactions t
286 WHERE t.txn_timestamp >= CURRENT_DATE - INTERVAL '90 days'
287     GROUP BY t.account_id
288 )
289 SELECT a.account_id,
290       c.full_name,
291       COALESCE(acc.n_txn, 0)           AS total_transaction,
292       COALESCE(txn.net_inflow_90d, 0) AS total_netflow
293   FROM accounts a
294   JOIN customers c ON c.customer_id = a.customer_id
295   LEFT JOIN acc ON acc.account_id = a.account_id
296   LEFT JOIN txn ON txn.account_id = a.account_id
297   ORDER BY a.account_id,
298           c.customer_id;
299
300 -- 17. Accounts with no transactions ever
301 SELECT c.customer_id, c.full_name, t.account_id, t.transaction_id
302   FROM customers c
303   JOIN accounts a ON c.customer_id = a.customer_id
304   JOIN transactions t ON a.account_id = t.account_id
305 WHERE t.transaction_id IS NULL
306   ORDER BY c.customer_id;
307
308 -- 18. Transactions per branch last month
309 SELECT b.branch_id, b.branch_name, COUNT(*) AS txn_count
310   FROM branches b
311   JOIN accounts a ON b.branch_id = a.branch_id
312   JOIN transactions t ON a.account_id = t.account_id
313 WHERE txn_timestamp >= DATE_TRUNC('Month', CURRENT_DATE) - INTERVAL '1 month'
314 AND txn_timestamp < DATE_TRUNC('Month', CURRENT_DATE)
315   GROUP BY b.branch_id, b.branch_name

```

```

316 ORDER BY b.branch_id;
317
318 -- 19. Days since last transaction per account (snapshot)
319 SELECT account_id,AGE(CURRENT_DATE,MAX(txn_timestamp)::DATE) AS days_since_last_transaction
320 FROM transactions
321 GROUP BY 1
322 ORDER BY 1;
323
324 -- 20. Outflow vs inflow by channel by branch (last 60 days)
325 SELECT b.branch_id,b.branch_name,
326     ROUND(SUM(CASE WHEN t.txn_type IN ('withdrawal','transfer_out','fee') THEN -t.amount ELSE 0 END),
2) AS outflows,
327     ROUND(SUM(CASE WHEN t.txn_type IN ('deposit','transfer_in','interest') THEN t.amount ELSE 0 END),
2) AS inflows
328     FROM branches b
329     JOIN accounts a ON b.branch_id=a.branch_id
330     JOIN transactions t ON a.account_id=t.account_id
331     WHERE t.txn_timestamp >= CURRENT_DATE - INTERVAL '60 days'
332     GROUP BY b.branch_id,b.branch_name
333     ORDER BY b.branch_id;
334
335
336 -- =====
337 -- 3) LOAN ANALYTICS
338 -- =====
339
340 -- 21. Loan count by type
341 SELECT loan_type,COUNT(*) AS ln_count
342 FROM loans
343 GROUP BY 1;
344
345 -- 22. Average interest rate by loan type
346 SELECT loan_type,ROUND(AVG(interest_rate),2) AS avg_intersest
347 FROM loans
348 GROUP BY 1;
349
350 -- 23. Loans started per quarter
351 SELECT DATE_TRUNC('quarter',start_date) AS str_qtr,COUNT(*) AS ln_count
352 FROM loans
353 GROUP BY 1
354 ORDER BY 1;
355
356 -- 24. EMIs per loan(simple interest)
357 SELECT * , ROUND((principal * interest_rate * (term_months/12)/100),2) AS EMI
358 FROM loans
359 ORDER BY loan_id;
360
361 -- 25. Total EMI & loans runs per customer
362 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
363     FROM accounts a
364     JOIN customers c ON a.customer_id=c.customer_id
365     JOIN loans l ON a.account_id=l.account_id
366     GROUP BY c.customer_id,c.full_name
367     ORDER BY c.customer_id;
368
369 -- 26. Customers that hasn't run any loan
370 SELECT c.customer_id,c.full_name
371     FROM customers c
372     JOIN accounts a ON c.customer_id=a.customer_id
373     LEFT JOIN loans l ON a.account_id=l.account_id
374     GROUP BY c.customer_id,c.full_name
375     HAVING COUNT(l.*)=0
376     ORDER BY c.customer_id;

```

```

377 -- 27. Top customers by total principal (across their loans)
378 SELECT c.customer_id,c.full_name,
379 SUM(l.principal) AS loan_principal
380 FROM customers c
381 JOIN accounts a ON c.customer_id=a.customer_id
382 JOIN loans l ON a.account_id=l.loan_id
383 GROUP BY c.customer_id,c.full_name
384 ORDER BY loan_principal DESC
385 LIMIT 50;
386
387
388 -- 28. Default rate by loan type
389 SELECT loan_type, ROUND(AVG(interest_rate),2) AS default_rate
390 FROM loans
391 WHERE status IN('defaulted')
392 GROUP BY loan_type;
393
394 -- 29. loan completion status
395 SELECT status, COUNT(*) AS no_of_customers
396 FROM loans
397 GROUP BY status
398 HAVING status='closed';
399
400
401 -- =====
402 -- 4) CARD & MERCHANT INSIGHT
403 -- =====
404
405 -- 30. Cards by network and status
406 SELECT network,status,COUNT(*) AS n_cards
407 FROM cards
408 GROUP BY network,status
409 ORDER BY network,status;
410
411 -- 31. Average credit limit by network
412 SELECT network,ROUND(AVG(credit_limit),2) AS avg_credit_limit
413 FROM cards
414 GROUP BY 1
415 ORDER BY 1;
416
417 -- 32. Purchases last month by merchant category
418 SELECT m.category,SUM(c.amount) AS total_purchase_amount
419 FROM merchants m
420 JOIN card_transactions c ON m.merchant_id=c.merchant_id
421 WHERE c.txn_timestamp>=DATE_TRUNC('Month',CURRENT_DATE) - INTERVAL '1 month'
422 AND c.txn_timestamp<DATE_TRUNC('Month',CURRENT_DATE)
423 AND c.txn_type='purchase'
424 GROUP BY m.category
425 ORDER BY total_purchase_amount DESC;
426
427 -- 33.TOP 50 customer by card uses when purchasing(All time)
428 SELECT c.customer_id,c.full_name,
429 COUNT(CASE WHEN ct.txn_type='purchase' THEN ct.card_txn_id ELSE 0 END) AS n_transactions,
430 SUM(CASE WHEN ct.txn_type='purchase' THEN ct.amount ELSE 0 END) AS total_spent
431 FROM customers c
432 JOIN accounts a ON c.customer_id = a.customer_id
433 JOIN cards ca ON a.account_id = ca.account_id
434 JOIN card_transactions ct ON ca.card_id = ct.card_id
435 GROUP BY c.customer_id,c.full_name
436 ORDER BY total_spent DESC
437 LIMIT 50;
438
439 -- 34.TOP 100 customer by card uses(in this year)
440 SELECT c.customer_id,c.full_name,

```

```

441 COUNT(CASE WHEN ct.txn_type='purchase' THEN ct.card_txn_id ELSE 0 END) AS n_transactions,
442 SUM(CASE WHEN ct.txn_type='purchase' THEN ct.amount ELSE 0 END) AS total_spent
443 FROM customers c
444 JOIN accounts a ON c.customer_id = a.customer_id
445 JOIN cards ca ON a.account_id = ca.account_id
446 JOIN card_transactions ct ON ca.card_id = ct.card_id
447 WHERE ct.txn_timestamp >= CURRENT_DATE - INTERVAL '1 year'
448 AND ct.txn_timestamp < '2025-01-01'
449 GROUP BY c.customer_id,c.full_name
450 ORDER BY total_spent DESC
451 LIMIT 100;
452
453 -- 35. Cash advance total per card (last 90 days)
454 SELECT card_id,
455 SUM(CASE WHEN txn_type='cash_advance' THEN amount ELSE 0 END) AS total_spent
456 FROM card_transactions
457 WHERE txn_timestamp >= CURRENT_DATE - INTERVAL '90 days'
458 GROUP BY card_id
459 ORDER BY total_spent DESC;
460
461 -- 36. Top 30 merchants by spend in last 60 days
462 SELECT m.merchant_id,m.merchant_name,m.category,
463 SUM(ct.amount) AS total_spent
464 FROM merchants m
465 JOIN card_transactions ct ON m.merchant_id = ct.merchant_id
466 WHERE ct.txn_timestamp >= CURRENT_DATE - INTERVAL '60 days'
467 GROUP BY m.merchant_id,m.merchant_name,m.category
468 ORDER BY total_spent DESC
469 LIMIT 30;
470
471
472
473
474 -- 37. Customers with cards but no card transactions
475 SELECT c.customer_id,c.full_name,COUNT(ct.*) AS n_transactions
476 FROM customers c
477 JOIN accounts a ON c.customer_id = a.customer_id
478 JOIN cards ca ON a.account_id = ca.account_id
479 LEFT JOIN card_transactions ct ON ca.card_id = ct.card_id
480 GROUP BY c.customer_id,c.full_name
481 HAVING COUNT(ct.*) IS NULL
482 ORDER BY c.customer_id,c.full_name;
483
484
485 =====
486 -- 5) CUSTOMER 360 VIEWS
487 =====
488
489 -- 38. Cardholders per customer
490 SELECT c.customer_id,c.full_name,COUNT(ca.*) AS n_cards
491 FROM customers c
492 LEFT JOIN accounts a ON c.customer_id = a.customer_id
493 LEFT JOIN cards ca ON a.account_id = ca.account_id
494 GROUP BY c.customer_id,c.full_name
495 ORDER BY c.customer_id,c.full_name;
496
497 -- 39. Total card spend per customer (lifetime)
498 SELECT c.customer_id,c.full_name,SUM(ct.amount) AS n_cards
499 FROM customers c
500 LEFT JOIN accounts a ON c.customer_id = a.customer_id
501 LEFT JOIN cards ca ON a.account_id = ca.account_id
502 LEFT JOIN card_transactions ct ON ca.card_id = ct.card_id
503 GROUP BY c.customer_id,c.full_name
504 HAVING SUM(ct.amount) IS NOT NULL

```

```

505 ORDER BY c.customer_id,c.full_name;
506
507 -- 40. Customers with any loan and at least one card
508 SELECT c.customer_id,c.full_name,
509 COUNT(l.*) AS n_loans,
510 COUNT(ca.*) AS n_cards
511 FROM customers c
512 LEFT JOIN accounts a ON c.customer_id = a.customer_id
513 LEFT JOIN loans l ON a.account_id = l.account_id
514 LEFT JOIN cards ca ON a.account_id = ca.account_id
515 GROUP BY c.customer_id,c.full_name
516 HAVING COUNT(ca.*) >= 1 AND COUNT(l.*) >=1
517 ORDER BY c.customer_id,c.full_name;
518
519 -- 41. Salary accounts by branch
520 SELECT b.branch_id,b.branch_name,
521 (COUNT(a.*)) FILTER(WHERE a.account_type = 'salary')) AS salary_accounts
522 FROM branches b
523 LEFT JOIN accounts a
524 ON b.branch_id = a.branch_id
525 GROUP BY b.branch_id,b.branch_name
526 ORDER BY b.branch_id,b.branch_name;
527
528
529 -- =====
530 -- 6) PERIOD & KPI SNAPSHOT
531 -- =====
532
533 -- 42. Interest credited YTD
534 SELECT (SUM(amount)) FILTER(WHERE txn_type = 'interest')) AS YTD_interest
535 FROM transactions
536 WHERE txn_timestamp>=DATE_TRUNC('Year',CURRENT_DATE);
537
538 -- 43. Accounts opened by year
539 SELECT EXTRACT(YEAR FROM opened_on) AS years,COUNT(*) n_accounts
540 FROM accounts
541 GROUP BY 1
542 ORDER BY 1;
543
544 -- 44. Total transactions amounts by year
545 SELECT EXTRACT(YEAR FROM txn_timestamp) AS years,
546 txn_type,
547 SUM(CASE WHEN txn_type IN ('fee','transfer_out','withdrawal') THEN -amount
548 ELSE amount END
549 )AS total_amount
550 FROM transactions
551 GROUP BY years,txn_type
552 ORDER BY years,txn_type;
553
554 -- 45. Customers opened accounts in this year
555 SELECT c.customer_id,c.full_name,
556 COUNT(a.*) n_accounts
557 FROM customers c
558 JOIN accounts a ON c.customer_id = a.customer_id
559 WHERE opened_on >= DATE_TRUNC('Year',CURRENT_DATE)
560 GROUP BY c.customer_id,c.full_name
561 ORDER BY c.customer_id,c.full_name;
562
563 -- 46. Total accounts opened YTD
564 SELECT SUM(n_accounts) AS total_YTD_accounts
565 FROM(
566 SELECT COUNT(a.account_id) AS n_accounts
567 FROM customers c
568 JOIN accounts a ON c.customer_id = a.customer_id

```

```

569 WHERE opened_on >= DATE_TRUNC('Year', CURRENT_DATE)
570 );
571
572 -- 47. Transactions completed by cards in this year
573 SELECT COUNT(*) AS YTD_transactions
574 FROM card_transactions
575 WHERE txn_timestamp >= DATE_TRUNC('Year', CURRENT_DATE);
576
577 -- 48. Customers that buys card in this year
578 SELECT c.customer_id, c.full_name, COUNT(ca.*) AS n_cards
579 FROM customers c
580 JOIN accounts a ON c.customer_id = a.customer_id
581 JOIN cards ca ON a.account_id = ca.account_id
582 WHERE ca.issued_on >= DATE_TRUNC('Year', CURRENT_DATE)
583 GROUP BY c.customer_id, c.full_name
584 ORDER BY c.customer_id, c.full_name;
585
586 -- 49. Cards expiring in next 60 days
587 SELECT card_id, account_id, network, expires_on
588 FROM cards
589 WHERE expires_on > CURRENT_DATE AND expires_on <= CURRENT_DATE + INTERVAL '60 days'
590 ORDER BY expires_on;
591
592
593 -- =====
594 -- 7) CUSTOMERS BALANCE SUMMARY
595 -- =====
596
597 -- 50. Cureent balance left in accounts by customers(who have accounts or cards or both)
598 WITH
599 acct AS (
600   SELECT customer_id, SUM(balance) AS sum_account_balance
601   FROM accounts
602   GROUP BY customer_id
603 ),
604 txns AS (
605   SELECT a.customer_id, SUM(t.amount) AS sum_txn_amount
606   FROM transactions t
607   JOIN accounts a ON t.account_id = a.account_id
608   GROUP BY a.customer_id
609 ),
610 card_txns AS (
611   SELECT a.customer_id, SUM(ct.amount) AS sum_card_txn
612   FROM cards ca
613   JOIN accounts a ON ca.account_id = a.account_id
614   JOIN card_transactions ct ON ca.card_id = ct.card_id
615   GROUP BY a.customer_id
616 )
617 SELECT
618   c.customer_id,
619   c.full_name,
620   COALESCE(ac.sum_account_balance, 0) AS total_account_balance,
621   COALESCE(tx.sum_txn_amount, 0) AS total_transactions,
622   COALESCE(cc.sum_card_txn, 0) AS total_card_transactions,
623   ROUND(
624     COALESCE(ac.sum_account_balance, 0)
625     + COALESCE(tx.sum_txn_amount, 0)
626     - COALESCE(cc.sum_card_txn, 0)
627   , 2) AS net_balance
628 FROM customers c
629 LEFT JOIN acct ac ON c.customer_id = ac.customer_id
630 LEFT JOIN txns tx ON c.customer_id = tx.customer_id
631 LEFT JOIN card_txns cc ON c.customer_id = cc.customer_id
632 -- optional: show only customers who have some accounts/cards/transactions

```

```

633 WHERE COALESCE(ac.sum_account_balance,0) <> 0
634     OR COALESCE(tx.sum_txn_amount,0) <> 0
635     OR COALESCE(cc.sum_card_txn,0) <> 0
636 ORDER BY c.customer_id;
637
638 -- 51. Cureent balance left in customers account id(who have accounts or cards or both)
639 SELECT
640     c.customer_id,
641     c.full_name,
642     a.account_id,
643     a.balance AS account_balance,
644     ROUND(
645         a.balance
646         + COALESCE(t.txn_sum,0)      -- transactions on this account
647         - COALESCE(ct.card_txn_sum,0) -- card transactions on cards linked to this account
648     , 2) AS net_balance
649 FROM customers c
650 JOIN accounts a ON c.customer_id = a.customer_id
651 LEFT JOIN (
652     SELECT account_id, SUM(amount) AS txn_sum
653     FROM transactions
654     GROUP BY account_id
655 ) t ON a.account_id = t.account_id
656 LEFT JOIN (
657     SELECT ca.account_id, SUM(ct.amount) AS card_txn_sum
658     FROM cards ca
659     JOIN card_transactions ct ON ca.card_id = ct.card_id
660     GROUP BY ca.account_id
661 ) ct ON a.account_id = ct.account_id
662 ORDER BY c.customer_id,
663     c.full_name;
664
665 -- 52. TOP 50 richest customers in our bank
666 WITH
667     acct AS (
668         SELECT customer_id, SUM(balance) AS sum_account_balance
669         FROM accounts
670         GROUP BY customer_id
671     ),
672     txns AS (
673         SELECT a.customer_id, SUM(t.amount) AS sum_txn_amount
674         FROM transactions t
675         JOIN accounts a ON t.account_id = a.account_id
676         GROUP BY a.customer_id
677     ),
678     card_txns AS (
679         SELECT a.customer_id, SUM(ct.amount) AS sum_card_txn
680         FROM cards ca
681         JOIN accounts a ON ca.account_id = a.account_id
682         JOIN card_transactions ct ON ca.card_id = ct.card_id
683         GROUP BY a.customer_id
684     )
685 SELECT
686     c.customer_id,
687     c.full_name,
688     COALESCE(ac.sum_account_balance,0) AS total_account_balance,
689     COALESCE(tx.sum_txn_amount,0)      AS total_transactions,
690     COALESCE(cc.sum_card_txn,0)       AS total_card_transactions,
691     ROUND(
692         COALESCE(ac.sum_account_balance,0)
693         + COALESCE(tx.sum_txn_amount,0)
694         - COALESCE(cc.sum_card_txn,0)
695     ,2) AS net_balance
696 FROM customers c

```

```

697 LEFT JOIN acct ac ON c.customer_id = ac.customer_id
698 LEFT JOIN txns tx ON c.customer_id = tx.customer_id
699 LEFT JOIN card_txns cc ON c.customer_id = cc.customer_id
700 WHERE COALESCE(ac.sum_account_balance,0) <> 0
701 ORDER BY net_balance DESC
702 LIMIT 50;
703
704 -- 53. Accounts that have low balance(Below 10000/-)
705 SELECT
706   c.customer_id,
707   c.full_name,
708   c.phone,
709   a.account_id,
710   a.balance AS account_balance,
711   ROUND(
712     a.balance
713     + COALESCE(t.txn_sum,0)      -- transactions on this account
714     - COALESCE(ct.card_txn_sum,0) -- card transactions on cards linked to this account
715   , 2) AS net_balance
716 FROM customers c
717 JOIN accounts a ON c.customer_id = a.customer_id
718 LEFT JOIN (
719   SELECT account_id, SUM(amount) AS txn_sum
720   FROM transactions
721   GROUP BY account_id
722 ) t ON a.account_id = t.account_id
723 LEFT JOIN (
724   SELECT ca.account_id, SUM(ct.amount) AS card_txn_sum
725   FROM cards ca
726   JOIN card_transactions ct ON ca.card_id = ct.card_id
727   GROUP BY ca.account_id
728 ) ct ON a.account_id = ct.account_id
729 WHERE (ROUND(
730   a.balance
731   + COALESCE(t.txn_sum,0)
732   - COALESCE(ct.card_txn_sum,0)
733   , 2)) <=10000
734 ORDER BY net_balance;
735
736
737 SELECT COUNT(account_id)
738 FROM accounts;

```



Filters

Overview

Branch Info.

Customers Info.

Cards

Loans

Customers

Total Customers

750



Total Accounts

1100



Total loans

400



Total Cards

800



Top 5 Customers

Aditya Gupta
₹4,61,976.56Krishna Chakraborty
₹5,00,345.67Navya Ghosh
₹4,92,346.47Navya Gupta
₹4,69,496.04Sai Verma
₹5,47,732.26

Treasury Value

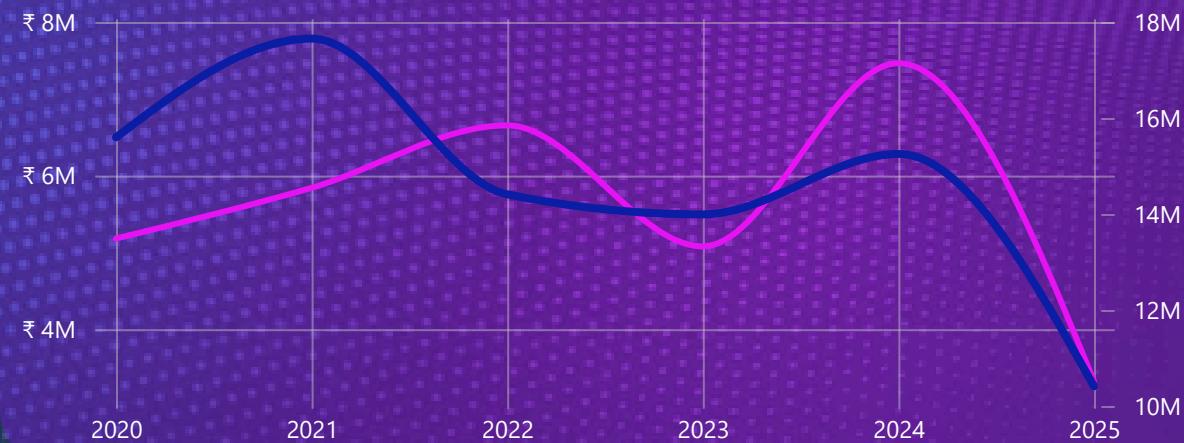
Current Treasury

₹ 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

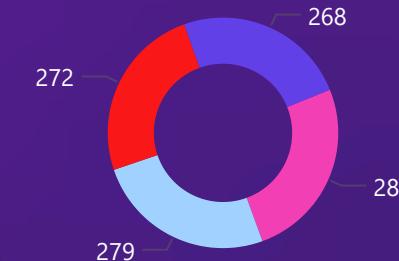
Bank charges

₹ 5,79,67,784...

8068

Accounts by Type

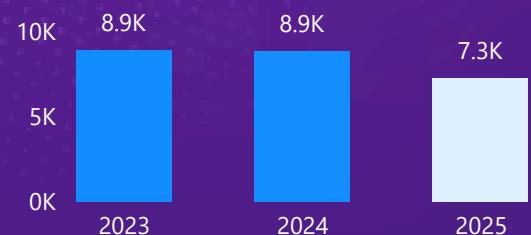
● salary ● fixed_dep... ● current



Transactions made by Year



Card Transactions made by Year



Overview

Branch Info.

Customers Info.

Cards

Loans

Customers

Account Balance

Mumbai 4 ...	3.0M	8.8M
Hyderabad...	4.0M	8.0M
Jaipur 6 Br...	3.3M	7.8M
Chennai 9 ...	2.1M	7.7M
Jaipur 3 Br...	2.6M	7.5M
Delhi 11 Br...	3.0M	7.2M
Mumbai 8 ...	1.4M	7.0M

Loans

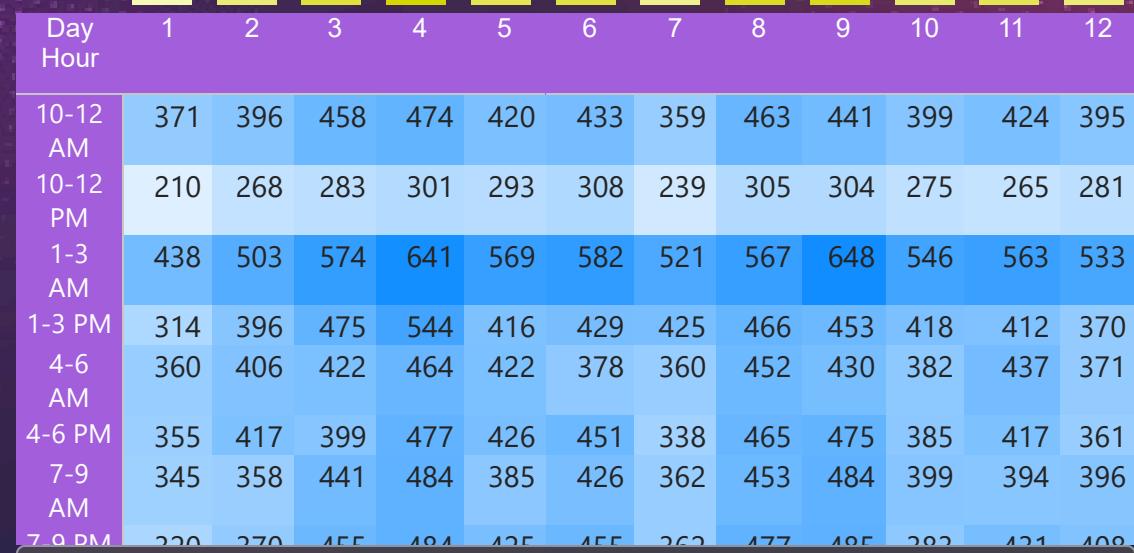
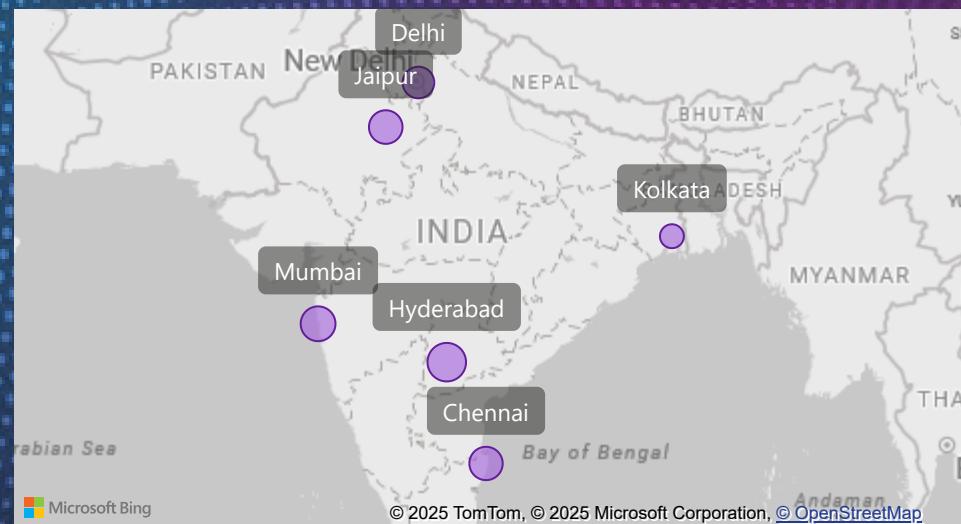
Mumbai 8 ...	₹32M
Jaipur 3 Br...	₹23M
Chennai 9 ...	₹22M
Mumbai 4 ...	₹19M
Delhi 11 Br...	₹19M
Hyderabad...	₹17M
Hyderabad...	₹16M

Card Transactions

Mumbai 4 ...	6.1M
Chennai 9 ...	5.8M
Mumbai 8 ...	5.2M
Jaipur 3 Br...	4.9M
Hyderabad...	4.5M
Jaipur 6 Br...	4.4M
Delhi 11 Br...	4.2M

Public Accounts

Mumbai...	107
Chennai...	102
Mumbai...	100
Jaipur 6...	96
Jaipur 3...	95
Hydera...	92
Delhi 1...	91



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
7	Hyderabad 7 Branch	65	29	5



Overview

Branch Info.

Customers Info.

Cards

Loans

Customers



49

Name

Aadhyaa Banerjee

D.O.B

01-28-1955

Email

aadhyaa.banerjee220@testmail.com

Ph No.

9002007565

City

Ahmedabad

KYC Status

pending

Account Created at

03-01-2020

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

customer_id

49

account_id

All

card_id

All

loan_id

All

No. of accounts

5

No. of Cards

2

No. of Loans

3

Transactions Summary

● atm ● branch ● imps ● mobile ● neft ● online ● rtgs ● upi



Card Info

Card ID 380

active credit

₹ 1,19,728.17

XXXX-XXXX-XXXX-2092

Mastercard 08-14-2027

Loan Info

Loan ID 65

active

gold

₹15,81,931.61

₹ 5,04,971.97

Months 96

30-03-2022

Card transactions

purchase

₹0.12M

cash_a...

₹0.01M

refund

0.00M₹

Bank charges

₹ 3,24,974.05



Account Balance



Overview

Branch Info.

Customers Info.

Cards

Loans

Customers

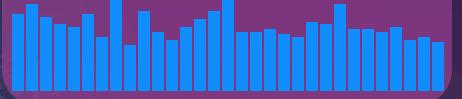
No of Card holders

585

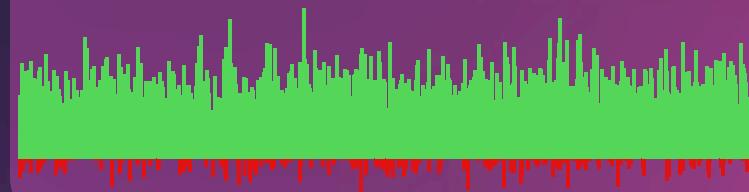


Total Cards

800



Transactions Analysis



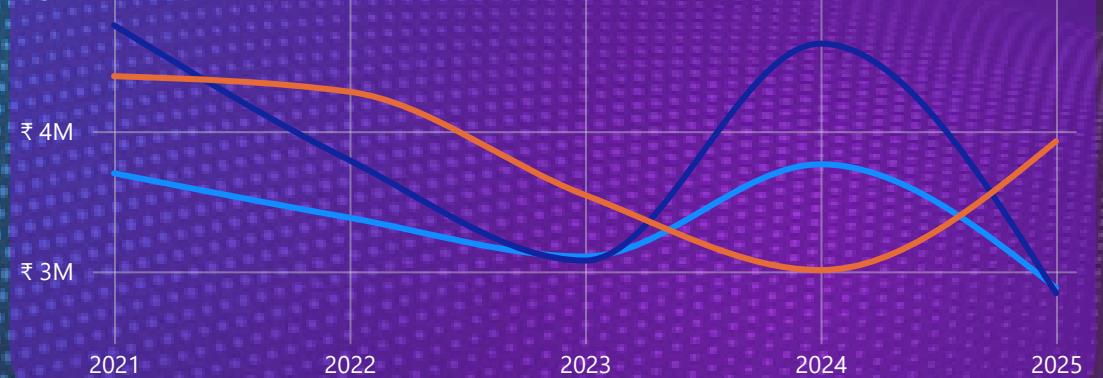
Transaction Timing Trend



Card transaction by Network

● Mastercard ● RuPay ● VISA

₹ 5M



Card transactions by Category

restaurants 4.7K

education 3.1K

fuel 3.0K

groceries 2.8K

utilities 2.6K

entertain... 2.4K

electronics 2.3K

fashion 1.4K

travel 1.3K

health 1.3K

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

Transaction amount by Merchants



Card type

RuPay 143

VISA 128

Mastercard 121

Card status

● active ● blocked ● expired

32.38% 34.38%

33.25%

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

Overview

Branch Info.

Customers Info.

Cards

Loans

Customers

No of Loan holders

331

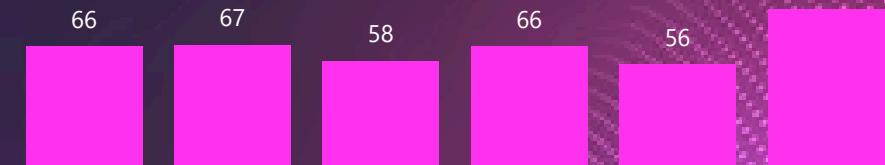
Loan Amount by Loan Type



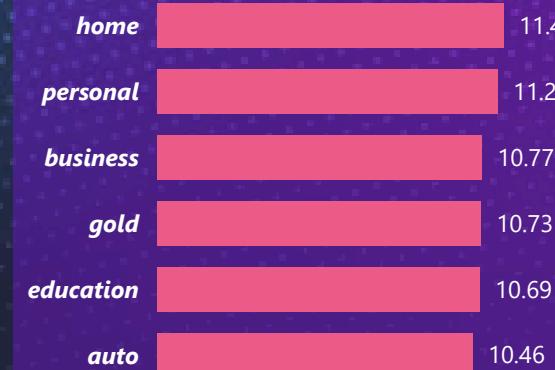
Total Loans

400

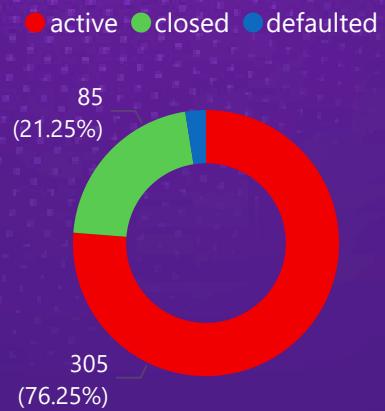
Loan Timing Trend



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



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	Ariun Chakraborty	2	1	2	₹2,32,036.48	₹2,76,617.92	₹ 44,362.83	₹6,42,827.97	₹ 7,74,014.70