

DIVYASHREE K

1BM19CS054

CSE-4A

Question:-Program 2:

Consider the following database for a banking enterprise. Branch (branch-name: String, branch-city: String, assets: real)

BankAccount(accno: int, branch-name: String,

balance: real) BankCustomer (customer-name:

String, customer-street: String, customer-city:

String) Depositer(customer-name: String,

accno: int) Loan (loan-number: int, branch-

name: String, amount: real) i. Create the above

tables by properly specifying the primary keys

and the foreign keys. ii. Enter at least five

tuples for each relation. iii. Find all the

customers who have at least two accounts at

the Main branch (ex. SBI_ResidencyRoad). iv.

Find all the customers who have an account at

all the branches located in a specific city (Ex.

Delhi). v. Demonstrate how you delete all

account tuples at every branch located in a

specific city (Ex. Bombay).

Program 2:

```
create database sample2;
```

```
use sample2;
```

```
CREATE TABLE branch
```

```
( branch_name VARCHAR(20),
```

```
branch_city VARCHAR(20),
```

```
assets REAL,
```

```
PRIMARY KEY(branch_name)
```

```
);
```

```
CREATE TABLE accounts
```

```
( acc_no INT,
```

```
branch_name VARCHAR(50),
```

```
balance REAL,  
PRIMARY KEY(acc_no),  
FOREIGN KEY(branch_name)  
REFERENCES branch(branch_name)  
ON UPDATE CASCADE ON DELETE  
CASCADE  
);
```

```
CREATE TABLE customer  
( customer_name VARCHAR(20),  
customer_street VARCHAR(50),  
customer_city VARCHAR(20),  
PRIMARY KEY(customer_name)  
);
```

```
CREATE TABLE depositor
( customer_name VARCHAR(20),
acc_no INT,
PRIMARY KEY(customer_name,
acc_no),
FOREIGN KEY(customer_name)
REFERENCES
customer(customer_name)
ON UPDATE CASCADE ON DELETE
CASCADE,
FOREIGN KEY(acc_no) REFERENCES
accounts(acc_no)
ON UPDATE CASCADE ON DELETE
CASCADE
);
```

```
CREATE TABLE loan
( loan_number INT,
  branch_name VARCHAR(50),
  amount REAL,
  PRIMARY KEY(loan_number),
  FOREIGN KEY(branch_name)
  REFERENCES branch(branch_name)
  ON UPDATE CASCADE ON DELETE
  CASCADE
);
```

```
INSERT INTO
branch(branch_name,branch_city,asset
s) VALUES
('SBI_Chamrajpet','Bangalore',50000),('
```

```
SBI_ResidencyRoad','Bangalore',10000)
,('SBI_ShivajiRoad','Bombay',20000),('S
BI_ParlimentRoad','Delhi',10000),('SBI_
Jantarmantar','Delhi',20000);
```

```
INSERT INTO
accounts(acc_no,branch_name,balance
) VALUES
(1,'SBI_Chamrajpet',2000),(2,'SBI_Resid
encyRoad',5000),(3,'SBI_ShivajiRoad',6
000),(4,'SBI_ParlimentRoad',9000),(5,'S
BI_Jantarmantar',8000),(6,'SBI_ShivajiR
oad',4000),(8,'SBI_ResidencyRoad',400
0),(9,'SBI_ParlimentRoad',3000),(10,'SB
I_ResidencyRoad',5000),(11,'SBI_Jantar
mantar',2000);
```

```
INSERT INTO
customer(customer_name,customer_street,customer_city) VALUES
('Avinash','Bull_Temple_Road','Bangalore'),('Dinesh','Bannerghatta_Road','Bangalore'),('Mohan','NationalCollege_Road','Bangalore'),('Nikil','Akbar_Road','Delhi'),('Ravi','Prithviraj_Road','Delhi');
```

```
INSERT INTO
depositor(customer_name,acc_no)
VALUES
('Avinash',1),('Dinesh',2),('Nikil',4),('Ravi',5),('Avinash',8),('Nikil',9),('Dinesh',10),('Nikil',11);
```

```
INSERT INTO
loan(loan_number,branch_name,amou
```

nt) VALUES

(1,'SBI_Chamrajpet',1000),(2,'SBI_ResidencyRoad',2000),(3,'SBI_ShivajiRoad',3000),(4,'SBI_ParlimentRoad',4000),(5,'SBI_Jantarmantar',5000);

SELECT * FROM branch;

SELECT * FROM accounts;

SELECT * FROM customer;

SELECT * FROM depositor;

SELECT * FROM loan;

SELECT * FROM customer WHERE
customer_name IN(SELECT
customer_name FROM depositor
group by customer_name having
COUNT(customer_name)>=2);


```
SELECT d.customer_name FROM
accounts a, depositor d,branch b
WHERE d.acc_no=a.acc_no AND
b.branch_name=a.branch_name AND
b.branch_city="Delhi" GROUP BY
d.customer_name having
count(distinct
b.branch_name)=(SELECT
COUNT(branch_name) FROM branch
WHERE branch_city="Delhi");
```

```
DELETE FROM ACCOUNTS WHERE
branch_name IN(SELECT branch_name
FROM BRANCH WHERE
branch_city='Bombay');
```

Output:

The screenshot shows a database management tool interface. At the top, there is a toolbar with various icons for file operations, editing, and execution. Below the toolbar, a list of SQL queries is displayed, each preceded by a line number and a bullet point. The queries are:

- 53 • `INSERT INTO depositor(customer_name, a`
- 54
- 55 • `INSERT INTO loan(loan_number, branch_n`
- 56
- 57 • `SELECT * FROM branch;`
- 58 • `SELECT * FROM accounts;`
- 59 • `SELECT * FROM customer;`

Below the queries, there is a horizontal scrollbar. Underneath the scrollbar, there is a section labeled "Result Grid" with a small icon of a grid. To the right of the "Result Grid" label is a "Filter Rows:" input field and an "Edit:" button. Below this section is a table displaying the results of the queries. The table has three columns: "branch_name", "branch_city", and "assets". The data rows are:

branch_name	branch_city	assets
SBI_Chamrajpet	Bangalore	50000
SBI_Jantarmanatar	Delhi	20000
SBI_ParliamentRoad	Delhi	10000
SBI_ResidencyRoad	Bangalore	10000
SBI_ShivajiRoad	Bombay	20000
NULL	NULL	NULL

54



55 • INSERT INTO loan(loan_number,bran

56

57 • SELECT * FROM branch;

58 • SELECT * FROM accounts;

<

Result Grid   Filter Rows: Ed

	acc_no	branch_name	balance
▶	1	SBI_Chamrajpet	2000
	2	SBI_ResidencyRoad	5000
	4	SBI_ParlimentRoad	9000
	5	SBI_Jantarmentar	8000
	8	SBI_ResidencyRoad	4000
	9	SBI_ParlimentRoad	3000
	10	SBI_ResidencyRoad	5000
	11	SBI_Jantarmentar	2000
*	TOTAL	TOTAL	TOTAL

54

55 • INSERT INTO loan(loan_number,branch_name,am

56




57 • SELECT * FROM branch;

58 • SELECT * FROM accounts;

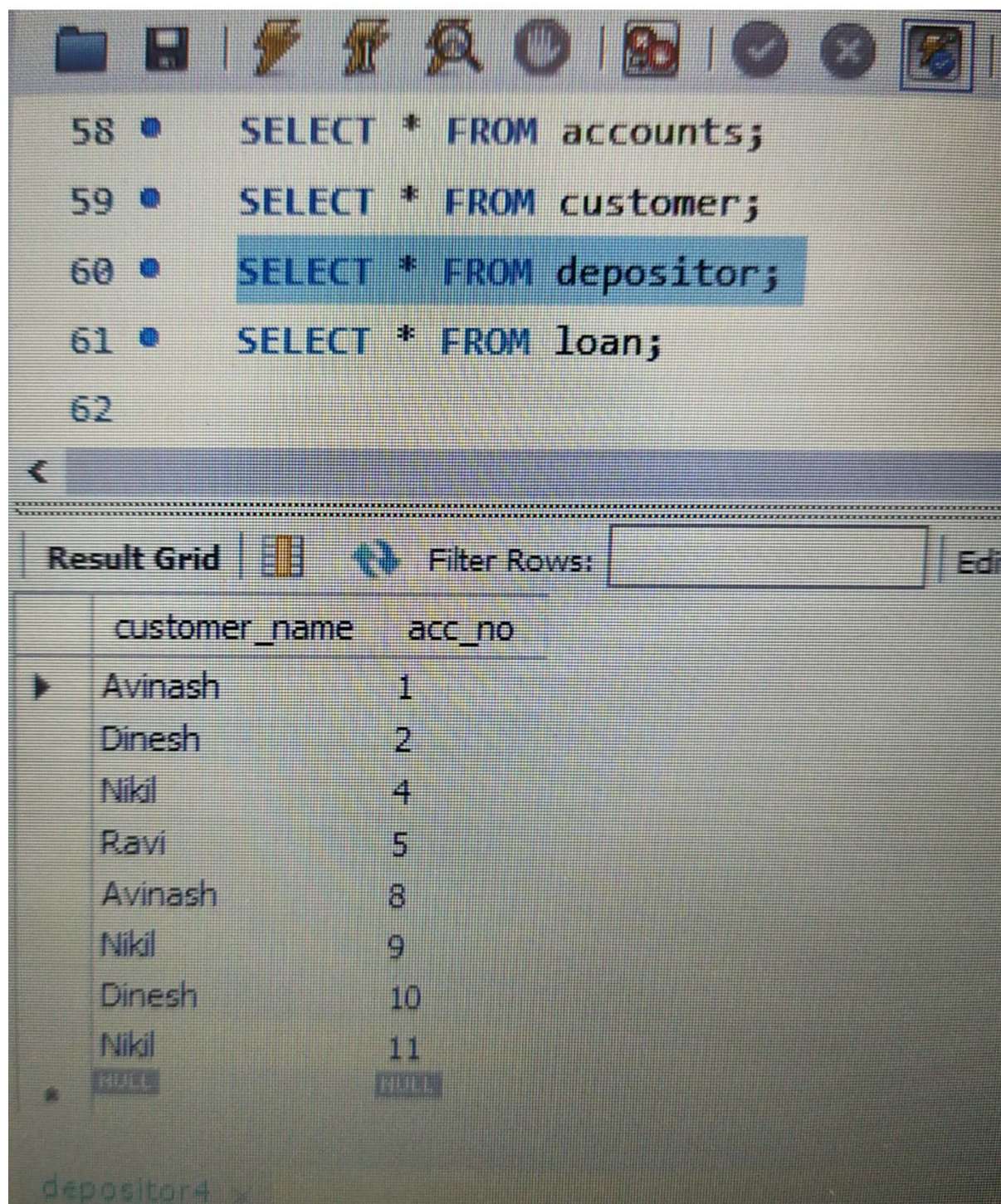
59 • SELECT * FROM customer;

60 • SELECT * FROM depositor;

<

Result Grid  Filter Rows: Edit:  



	customer_name	customer_street	customer_city
▶	Avinash	Bull_Temple_Road	Bangalore
	Dinesh	Bannergatta_Road	Bangalore
	Mohan	NationalCollege_Road	Bangalore
	Nikil	Akbar_Road	Delhi
	Ravi	Prithviraj_Road	Delhi
•	NULL	NULL	NULL



The screenshot shows a database query editor with a toolbar at the top containing icons for file operations, execution, and navigation. Below the toolbar, a list of SQL queries is displayed, each preceded by a line number. The query at line 60, `SELECT * FROM depositor;`, is highlighted in blue. Below the queries, a horizontal bar contains a back arrow and a search input field. The results of the highlighted query are shown in a table with the title "Result Grid". The table has two columns: "customer_name" and "acc_no". It contains 11 rows of data, including names like Avinash, Dinesh, Nikil, and Ravi, along with their respective account numbers. The last row shows "NULL" for both fields. At the bottom left, the text "depositor4" is visible.

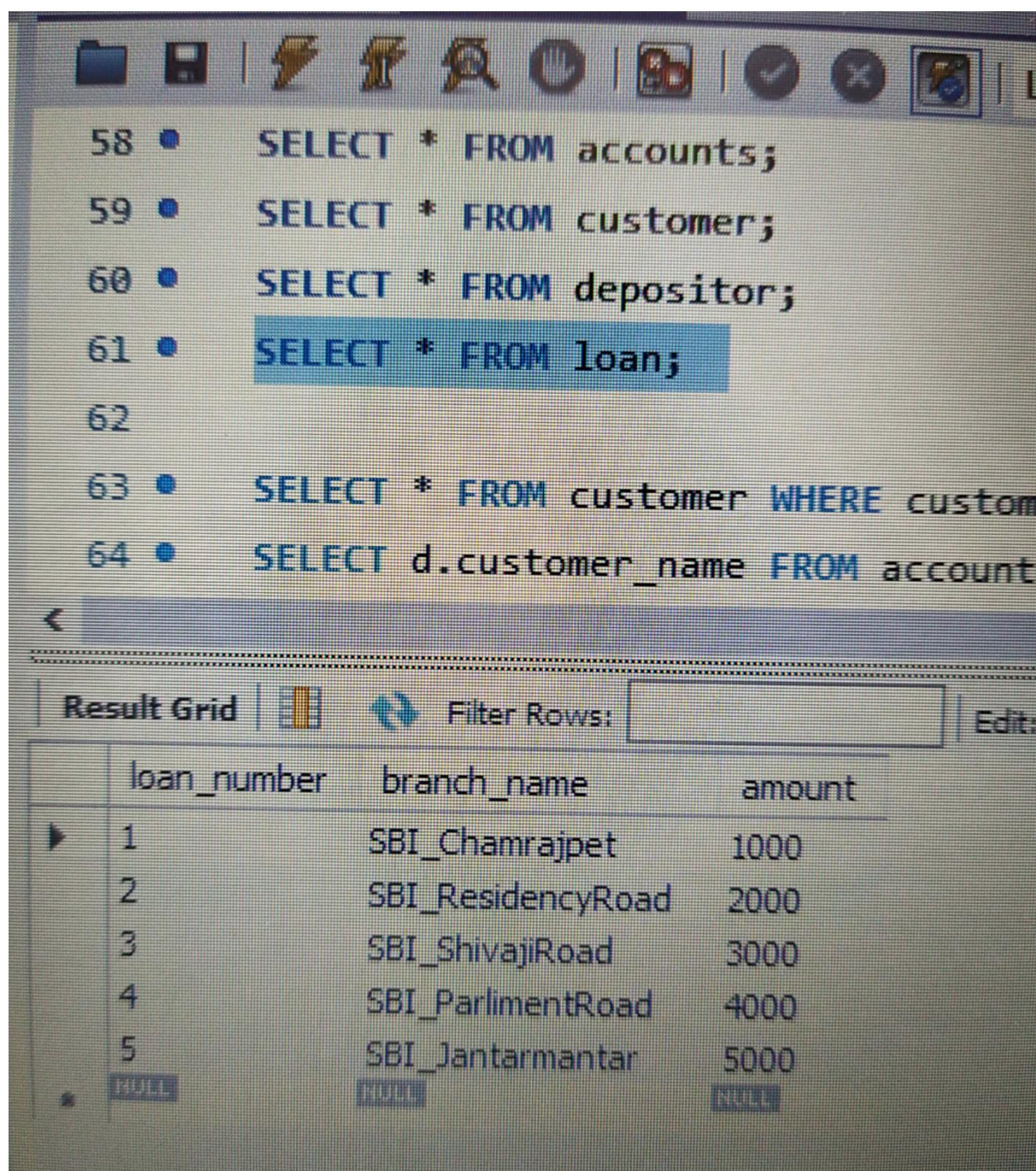
```
58 SELECT * FROM accounts;
59 SELECT * FROM customer;
60 SELECT * FROM depositor;
61 SELECT * FROM loan;
62
```

<

Result Grid   Filter Rows: Edit

	customer_name	acc_no
▶	Avinash	1
	Dinesh	2
	Nikil	4
	Ravi	5
	Avinash	8
	Nikil	9
	Dinesh	10
	Nikil	11
*	NULL	NULL


depositor4 x



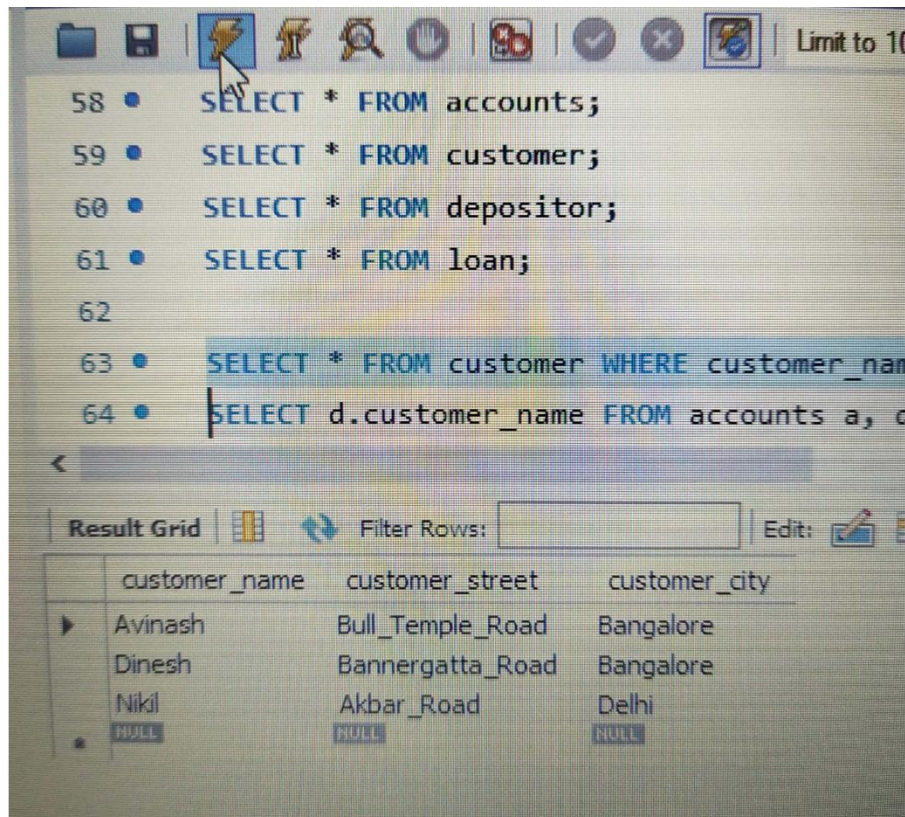
The screenshot shows a database query editor with a toolbar at the top containing icons for file operations, execution, and navigation. Below the toolbar, a list of SQL queries is displayed, each preceded by a line number and a bullet point. The query on line 61, "SELECT * FROM loan;", is highlighted with a blue background. Below the queries, a "Result Grid" section is visible, featuring a "Filter Rows:" input field and an "Edit:" button. The result grid itself is a table with four columns: "loan_number", "branch_name", and "amount". It contains five rows of data, with the last row showing "NULL" values for all three columns.

```
58 • SELECT * FROM accounts;
59 • SELECT * FROM customer;
60 • SELECT * FROM depositor;
61 • SELECT * FROM loan;
62
63 • SELECT * FROM customer WHERE custom
64 • SELECT d.customer_name FROM account
```

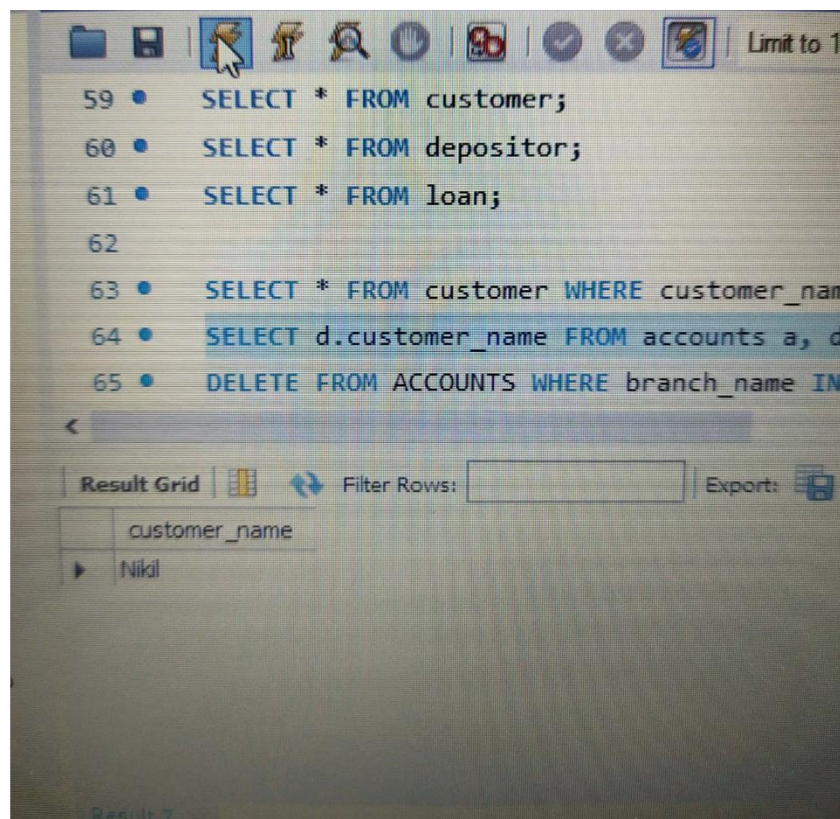
<

Result Grid |  Filter Rows: Edit:

	loan_number	branch_name	amount
▶	1	SBI_Chamrajpet	1000
	2	SBI_ResidencyRoad	2000
	3	SBI_ShivajiRoad	3000
	4	SBI_ParliamentRoad	4000
	5	SBI_Jantarmantar	5000
*	NULL	NULL	NULL



Scanned with CamScanner



Scanned with CamScanner