NAME:	Shubham Golwal
UID NO.	2020300015
Experiment No.	03

AIM:	Vertical fragmentation
Program 1	
PROBLEM STATEMENT :	Write the scenario of any application in distributed database and consider, one relation from that and fragment that relation into vertical fragmentation and execute the queries on that fragment
THEORY :	**What is Fragmentation?** Fragmentation in ADBMS (Advanced Database Management Systems) refers to the process of dividing a large database into smaller and more manageable parts, called fragments. This is done to improve the performance, scalability, and availability of the database by distributing the data across multiple servers, disks, or storage devices. This can also help in reducing the size of individual fragments and improve the access time for specific data subsets, making it easier to manage and maintain the database. There are two main types of fragmentation in ADBMS: 1. Horizontal Fragmentation 2. Vertical Fragmentation **Vertical Fragmentation** • Vertical fragmentation is a database design technique used to divide a large table into smaller tables based on their columns. This technique helps to improve query performance by reducing the amount of data that needs to be scanned by the database engine. • By breaking a large table into smaller tables, the database engine can scan a smaller set of data to answer a query, which can result in faster query performance. • Vertical fragmentation can also improve data management by separating related columns into smaller, more manageable tables. • If a particular column is frequently queried, it can be placed in a separate table to improve performance. Alternatively, if a column is

rarely used, it can be placed in a separate table to reduce the size of the main table and improve performance.

Advantages of fragmentation:

Before we discuss fragmentation in detail, we list four reasons for fragmenting a relation

Usage

In general, applications work with views rather than entire relations. Therefore, for data distribution, it seems appropriate to work with subsets of relation as the unit of distribution.

Efficiency

Data is stored close to where it is most frequently used. In addition, data that is ‚not needed by’ local applications is not stored.

Parallelism

With fragments as the unit of distribution, a transaction can be divided into several sub queries that operate on fragments. This should increase the degree of concurrency, or parallelism, in the system, thereby allowing transactions that can do so safely to execute in parallel.

Security

Data not required by local applications is not stored, and consequently not available to unauthorized users.

Disadvantages of fragmentation

Fragmentation has two primary disadvantages, which we have mentioned previously:

Performance

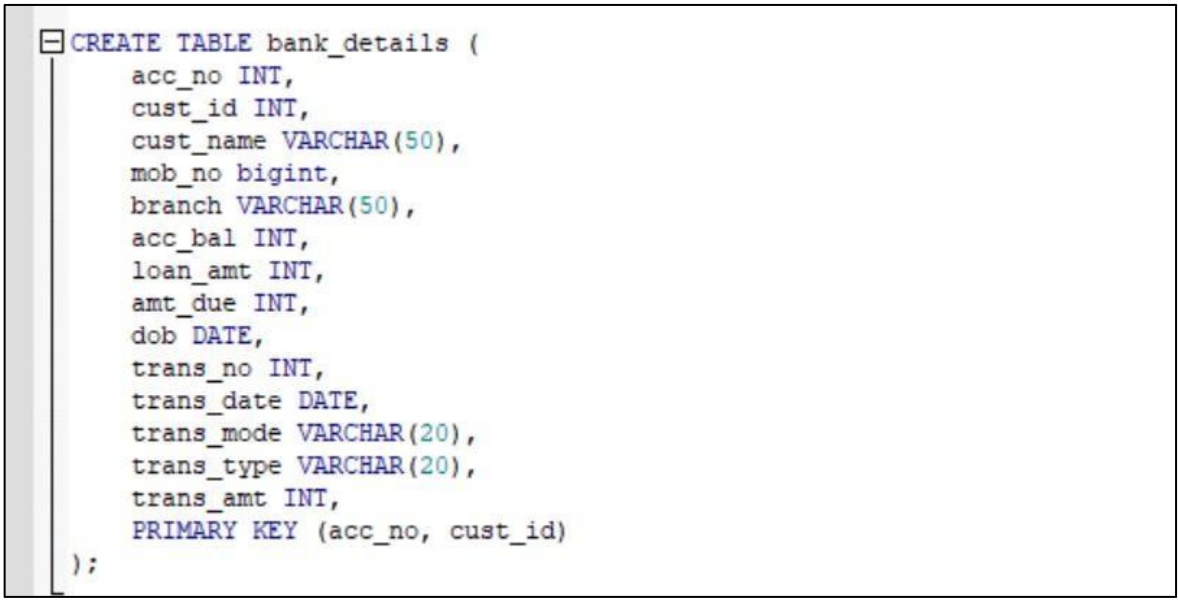
The performance of global application that requires data from several fragments located at different sites may be slower.

Integrity

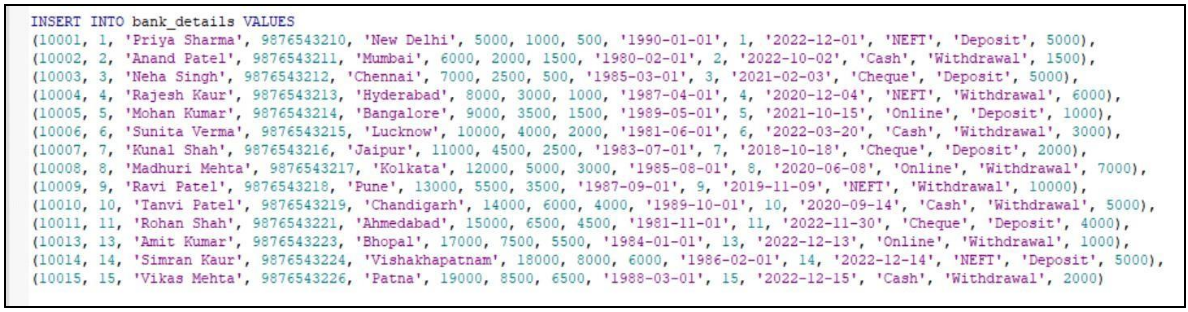
Integrity control may be more difficult if data and functional dependencies are fragmented and located at different sites.

OUTPUT:

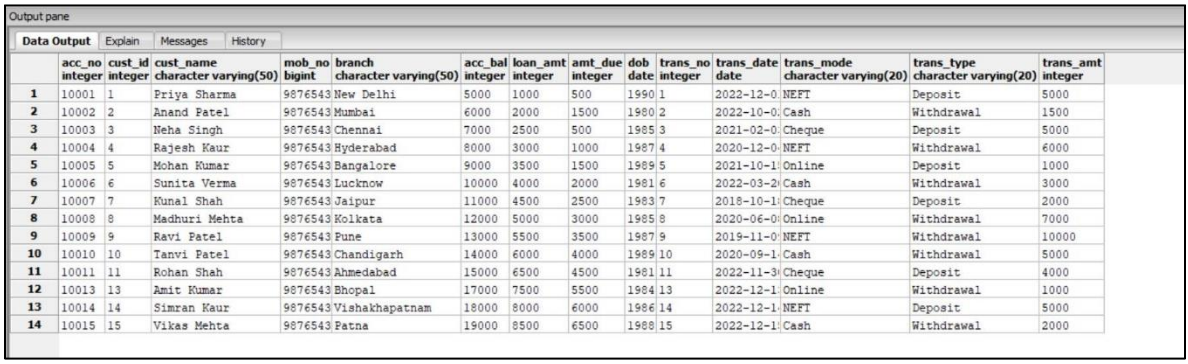
Creation and data insertion in bank details table :



```
CREATE TABLE bank_details (
    acc_no INT,
    cust_id INT,
    cust_name VARCHAR(50),
    mob_no bigint,
    branch VARCHAR(50),
    acc_bal INT,
    loan_amt INT,
    amt_due INT,
    dob DATE,
    trans_no INT,
    trans_date DATE,
    trans_mode VARCHAR(20),
    trans_type VARCHAR(20),
    trans_amt INT,
    PRIMARY KEY (acc_no, cust_id)
);
```



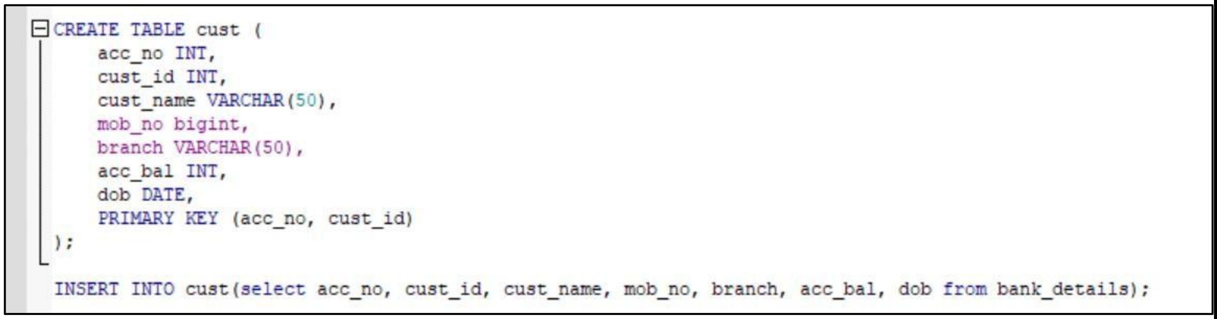
```
INSERT INTO bank_details VALUES
(10001, 1, 'Priya Sharma', 9876543210, 'New Delhi', 5000, 1000, 500, '1990-01-01', 1, '2022-12-01', 'NEFT', 'Deposit', 5000),
(10002, 2, 'Anand Patel', 9876543211, 'Mumbai', 6000, 2000, 1500, '1980-02-01', 2, '2022-10-02', 'Cash', 'Withdrawal', 1500),
(10003, 3, 'Neha Singh', 9876543212, 'Chennai', 7000, 2500, 500, '1985-03-01', 3, '2021-02-03', 'Cheque', 'Deposit', 5000),
(10004, 4, 'Rajesh Kaur', 9876543213, 'Hyderabad', 8000, 3000, 1000, '1987-04-01', 4, '2020-12-04', 'NEFT', 'Withdrawal', 6000),
(10005, 5, 'Mohan Kumar', 9876543214, 'Bangalore', 9000, 3500, 1500, '1989-05-01', 5, '2021-10-15', 'Online', 'Deposit', 1000),
(10006, 6, 'Sunita Verma', 9876543215, 'Lucknow', 10000, 4000, 2000, '1981-06-01', 6, '2022-03-20', 'Cash', 'Withdrawal', 3000),
(10007, 7, 'Kunal Shah', 9876543216, 'Jaipur', 11000, 4500, 2500, '1983-07-01', 7, '2018-10-18', 'Cheque', 'Deposit', 2000),
(10008, 8, 'Madhuri Mehta', 9876543217, 'Kolkata', 12000, 5000, 3000, '1985-08-01', 8, '2020-06-08', 'Online', 'Withdrawal', 7000),
(10009, 9, 'Ravi Patel', 9876543218, 'Pune', 13000, 5500, 3500, '1987-09-01', 9, '2019-11-09', 'NEFT', 'Withdrawal', 10000),
(10010, 10, 'Tanvi Patel', 9876543219, 'Chandigarh', 14000, 6000, 4000, '1989-10-01', 10, '2020-09-14', 'Cash', 'Withdrawal', 5000),
(10011, 11, 'Rohan Shah', 9876543221, 'Ahmedabad', 15000, 6500, 4500, '1981-11-01', 11, '2022-11-30', 'Cheque', 'Deposit', 4000),
(10013, 13, 'Amit Kumar', 9876543223, 'Bhopal', 17000, 7500, 5500, '1984-01-01', 13, '2022-12-13', 'Online', 'Withdrawal', 1000),
(10014, 14, 'Simran Kaur', 9876543224, 'Vishakhapatnam', 18000, 8000, 6000, '1986-02-01', 14, '2022-12-14', 'NEFT', 'Deposit', 5000),
(10015, 15, 'Vikas Mehta', 9876543226, 'Patna', 19000, 8500, 6500, '1988-03-01', 15, '2022-12-15', 'Cash', 'Withdrawal', 2000)
```



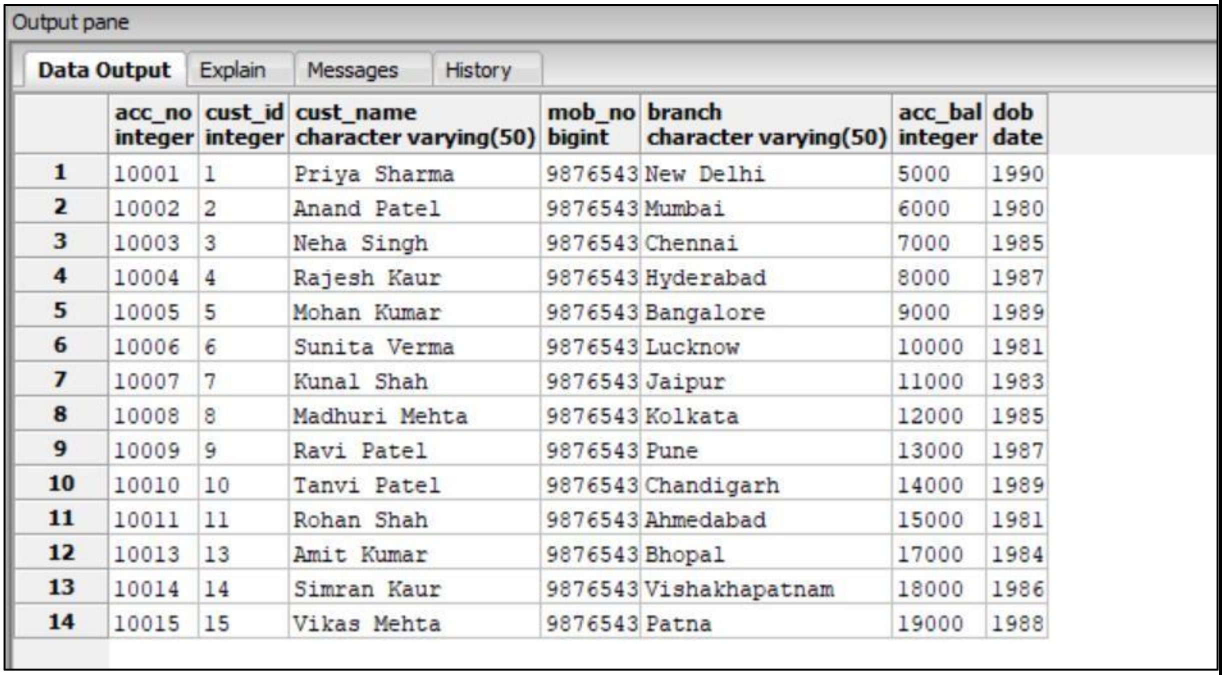
	acc_no	cust_id	cust_name	mob_no	branch	acc_bal	loan_amt	amt_due	dob	trans_no	trans_date	trans_mode	trans_type	trans_amt
	integer	integer	character varying(50)	bigint	character varying(50)	integer	integer	integer	date	integer	date	character varying(20)	character varying(20)	integer
1	10001	1	Priya Sharma	9876543	New Delhi	5000	1000	500	1990	1	2022-12-0	NEFT	Deposit	5000
2	10002	2	Anand Patel	9876543	Mumbai	6000	2000	1500	1980	2	2022-10-0	Cash	Withdrawal	1500
3	10003	3	Neha Singh	9876543	Chennai	7000	2500	500	1985	3	2021-02-0	Cheque	Deposit	5000
4	10004	4	Rajesh Kaur	9876543	Hyderabad	8000	3000	1000	1987	4	2020-12-0	NEFT	Withdrawal	6000
5	10005	5	Mohan Kumar	9876543	Bangalore	9000	3500	1500	1989	5	2021-10-1	Online	Deposit	1000
6	10006	6	Sunita Verma	9876543	Lucknow	10000	4000	2000	1981	6	2022-03-2	Cash	Withdrawal	3000
7	10007	7	Kunal Shah	9876543	Jaipur	11000	4500	2500	1983	7	2018-10-1	Cheque	Deposit	2000
8	10008	8	Madhuri Mehta	9876543	Kolkata	12000	5000	3000	1985	8	2020-06-0	Online	Withdrawal	7000
9	10009	9	Ravi Patel	9876543	Pune	13000	5500	3500	1987	9	2019-11-0	NEFT	Withdrawal	10000
10	10010	10	Tanvi Patel	9876543	Chandigarh	14000	6000	4000	1989	10	2020-09-1	Cash	Withdrawal	5000
11	10011	11	Rohan Shah	9876543	Ahmedabad	15000	6500	4500	1981	11	2022-11-3	Cheque	Deposit	4000
12	10013	13	Amit Kumar	9876543	Bhopal	17000	7500	5500	1984	13	2022-12-1	Online	Withdrawal	1000
13	10014	14	Simran Kaur	9876543	Vishakhapatnam	18000	8000	6000	1986	14	2022-12-1	NEFT	Deposit	5000
14	10015	15	Vikas Mehta	9876543	Patna	19000	8500	6500	1988	15	2022-12-1	Cash	Withdrawal	2000

Vertical Fragmentation of bank details table

Creation and insertion of data in cust table:

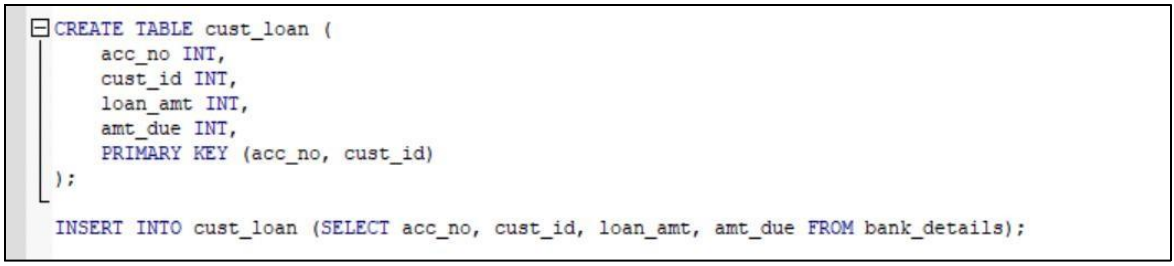


```
CREATE TABLE cust (
    acc_no INT,
    cust_id INT,
    cust_name VARCHAR(50),
    mob_no bigint,
    branch VARCHAR(50),
    acc_bal INT,
    dob DATE,
    PRIMARY KEY (acc_no, cust_id)
);

INSERT INTO cust(select acc_no, cust_id, cust_name, mob_no, branch, acc_bal, dob from bank_details);
```

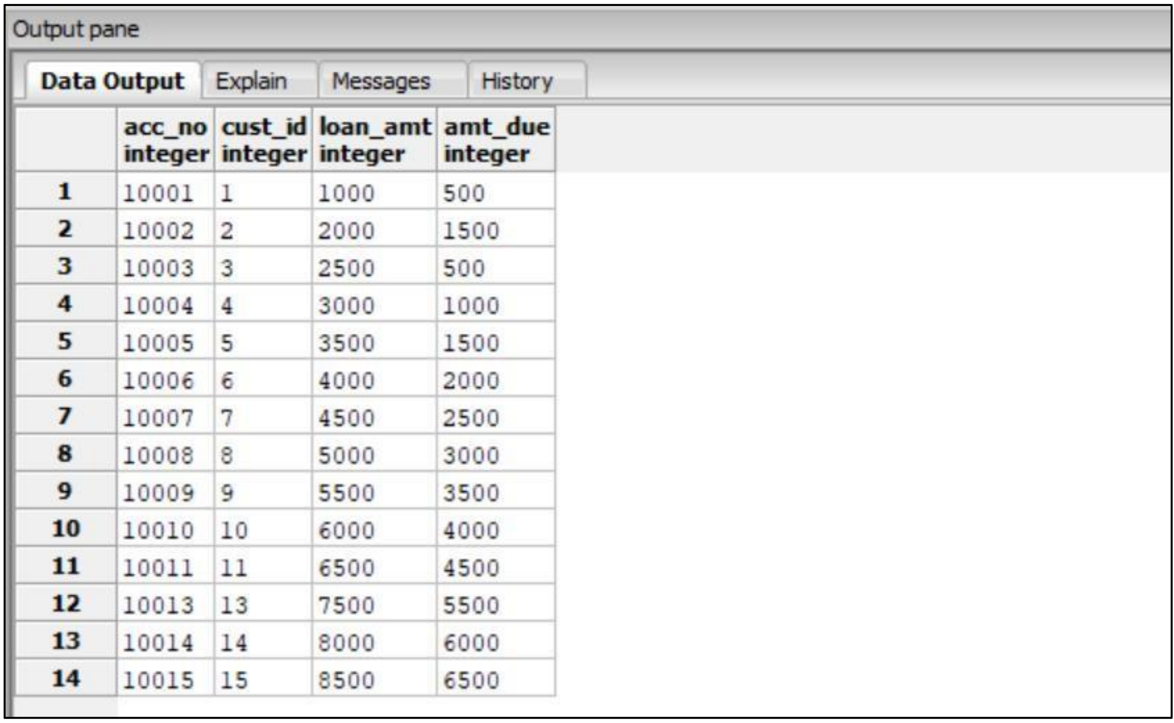


	acc_no integer	cust_id integer	cust_name character varying(50)	mob_no bigint	branch character varying(50)	acc_bal integer	dob date
1	10001	1	Priya Sharma	9876543	New Delhi	5000	1990
2	10002	2	Anand Patel	9876543	Mumbai	6000	1980
3	10003	3	Neha Singh	9876543	Chennai	7000	1985
4	10004	4	Rajesh Kaur	9876543	Hyderabad	8000	1987
5	10005	5	Mohan Kumar	9876543	Bangalore	9000	1989
6	10006	6	Sunita Verma	9876543	Lucknow	10000	1981
7	10007	7	Kunal Shah	9876543	Jaipur	11000	1983
8	10008	8	Madhuri Mehta	9876543	Kolkata	12000	1985
9	10009	9	Ravi Patel	9876543	Pune	13000	1987
10	10010	10	Tanvi Patel	9876543	Chandigarh	14000	1989
11	10011	11	Rohan Shah	9876543	Ahmedabad	15000	1981
12	10013	13	Amit Kumar	9876543	Bhopal	17000	1984
13	10014	14	Simran Kaur	9876543	Vishakhapatnam	18000	1986
14	10015	15	Vikas Mehta	9876543	Patna	19000	1988

Creation and insertion of data in cust_loan table :

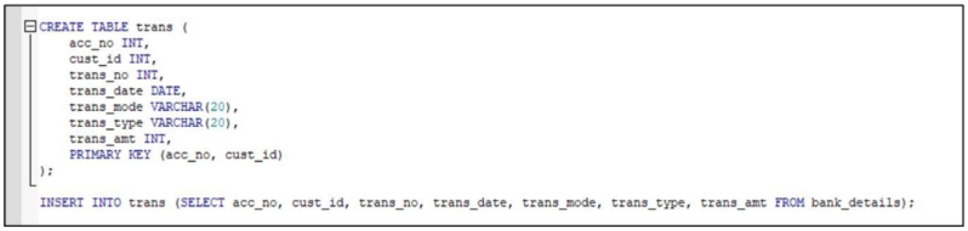


```
CREATE TABLE cust_loan (
    acc_no INT,
    cust_id INT,
    loan_amt INT,
    amt_due INT,
    PRIMARY KEY (acc_no, cust_id)
);

INSERT INTO cust_loan (SELECT acc_no, cust_id, loan_amt, amt_due FROM bank_details);
```

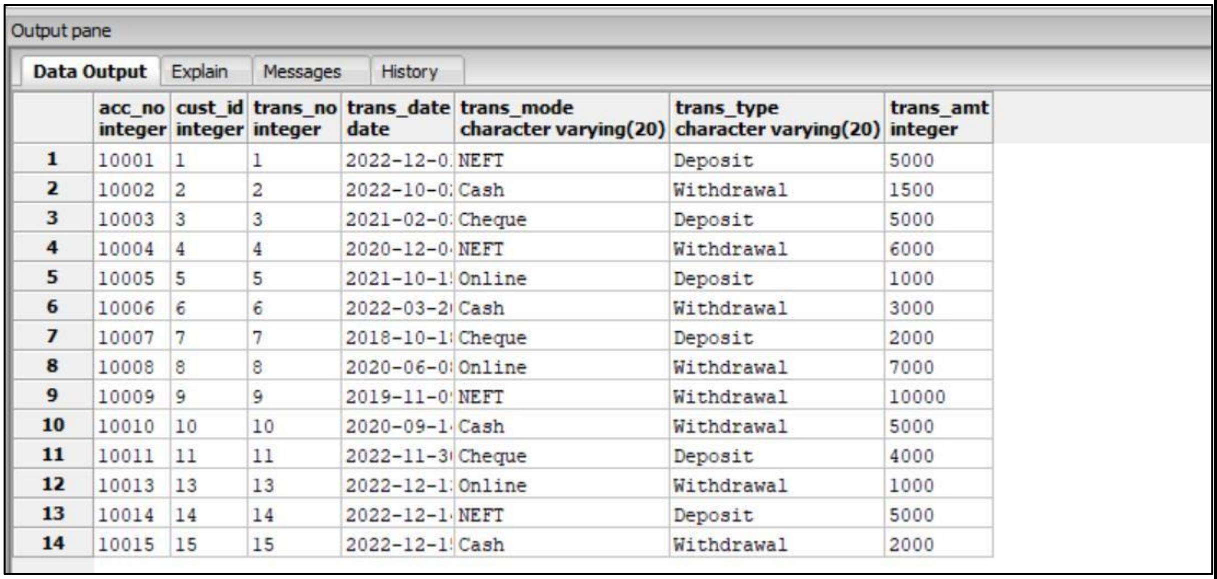


	acc_no integer	cust_id integer	loan_amt integer	amt_due integer
1	10001	1	1000	500
2	10002	2	2000	1500
3	10003	3	2500	500
4	10004	4	3000	1000
5	10005	5	3500	1500
6	10006	6	4000	2000
7	10007	7	4500	2500
8	10008	8	5000	3000
9	10009	9	5500	3500
10	10010	10	6000	4000
11	10011	11	6500	4500
12	10013	13	7500	5500
13	10014	14	8000	6000
14	10015	15	8500	6500

Creation and insertion of data in transaction table:



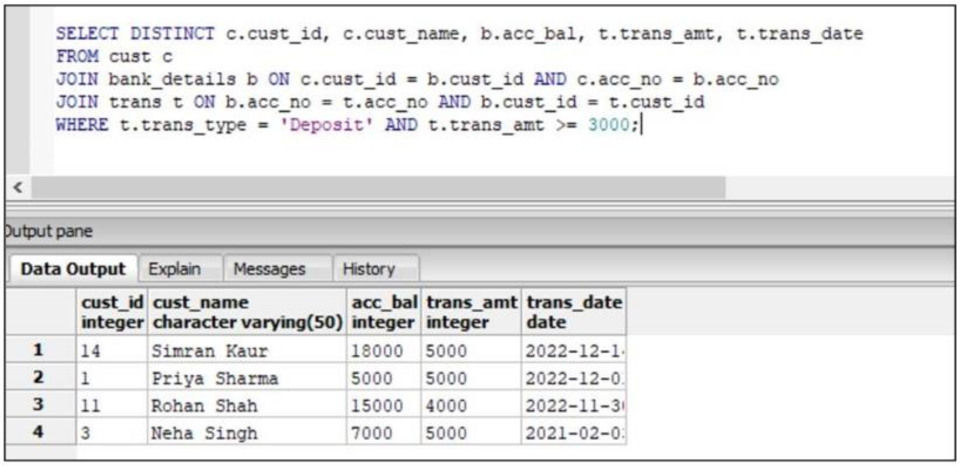
```
CREATE TABLE trans (
    acc_no INT,
    cust_id INT,
    trans_no INT,
    trans_date DATE,
    trans_mode VARCHAR(20),
    trans_type VARCHAR(20),
    trans_amt INT,
    PRIMARY KEY (acc_no, cust_id)
);

INSERT INTO trans (SELECT acc_no, cust_id, trans_no, trans_date, trans_mode, trans_type, trans_amt FROM bank_details);
```



	acc_no integer	cust_id integer	trans_no integer	trans_date date	trans_mode character varying(20)	trans_type character varying(20)	trans_amt integer
1	10001	1	1	2022-12-0	NEFT	Deposit	5000
2	10002	2	2	2022-10-0	Cash	Withdrawal	1500
3	10003	3	3	2021-02-0	Cheque	Deposit	5000
4	10004	4	4	2020-12-0	NEFT	Withdrawal	6000
5	10005	5	5	2021-10-1	Online	Deposit	1000
6	10006	6	6	2022-03-2	Cash	Withdrawal	3000
7	10007	7	7	2018-10-1	Cheque	Deposit	2000
8	10008	8	8	2020-06-0	Online	Withdrawal	7000
9	10009	9	9	2019-11-0	NEFT	Withdrawal	10000
10	10010	10	10	2020-09-1	Cash	Withdrawal	5000
11	10011	11	11	2022-11-3	Cheque	Deposit	4000
12	10013	13	13	2022-12-1	Online	Withdrawal	1000
13	10014	14	14	2022-12-1	NEFT	Deposit	5000
14	10015	15	15	2022-12-1	Cash	Withdrawal	2000

Observations on Fragmented Tables:

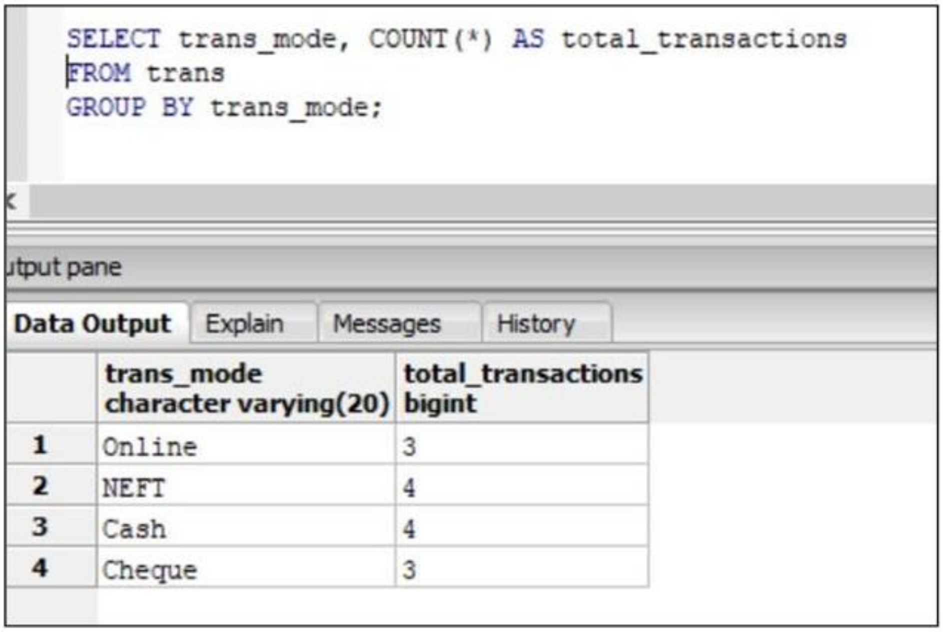
1. Retrieve the details of all customers who have made a deposit of more than or equal to 3000



```sql
SELECT DISTINCT c.cust_id, c.cust_name, b.acc_bal, t.trans_amt, t.trans_date
FROM cust c
JOIN bank_details b ON c.cust_id = b.cust_id AND c.acc_no = b.acc_no
JOIN trans t ON b.acc_no = t.acc_no AND b.cust_id = t.cust_id
WHERE t.trans_type = 'Deposit' AND t.trans_amt >= 3000;
```

	cust_id integer	cust_name character varying(50)	acc_bal integer	trans_amt integer	trans_date date
1	14	Simran Kaur	18000	5000	2022-12-1
2	1	Priya Sharma	5000	5000	2022-12-0
3	11	Rohan Shah	15000	4000	2022-11-3
4	3	Neha Singh	7000	5000	2021-02-0

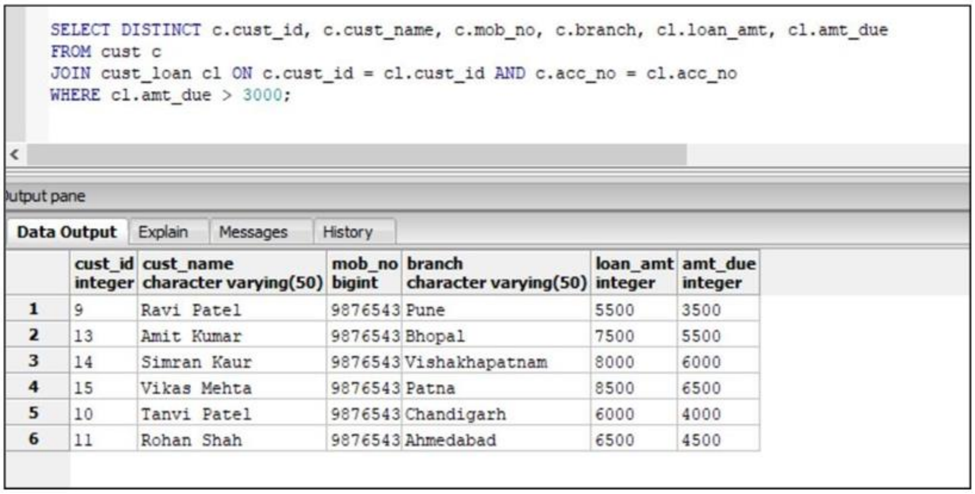
2. Retrieve the total number of transactions made via each mode of transaction



```
SELECT trans_mode, COUNT(*) AS total_transactions
FROM trans
GROUP BY trans_mode;
```

	trans_mode character varying(20)	total_transactions bigint
1	Online	3
2	NEFT	4
3	Cash	4
4	Cheque	3

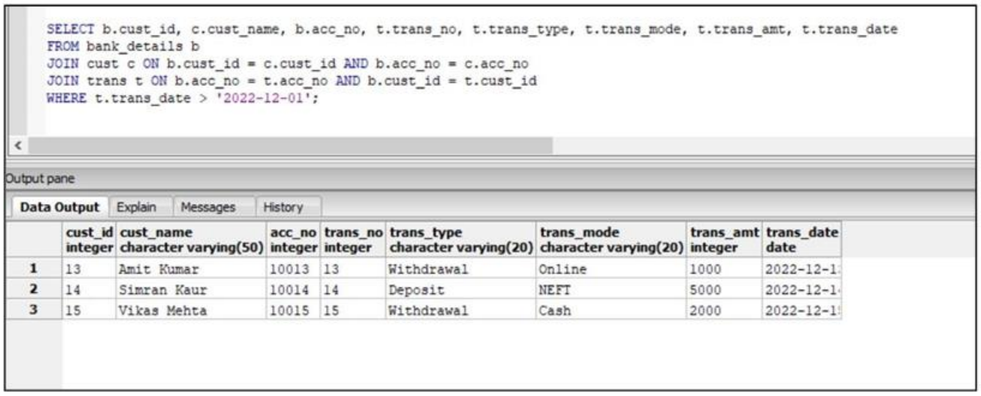
3. Retrieve the details of all customers with due loan amount greater than 3000



```
SELECT DISTINCT c.cust_id, c.cust_name, c.mob_no, c.branch, cl.loan_amt, cl.amt_due
FROM cust c
JOIN cust_loan cl ON c.cust_id = cl.cust_id AND c.acc_no = cl.acc_no
WHERE cl.amt_due > 3000;
```

	cust_id integer	cust_name character varying(50)	mob_no bigint	branch character varying(50)	loan_amt integer	amt_due integer
1	9	Ravi Patel	9876543	Pune	5500	3500
2	13	Amit Kumar	9876543	Bhopal	7500	5500
3	14	Simran Kaur	9876543	Vishakhapatnam	8000	6000
4	15	Vikas Mehta	9876543	Patna	8500	6500
5	10	Tanvi Patel	9876543	Chandigarh	6000	4000
6	11	Rohan Shah	9876543	Ahmedabad	6500	4500

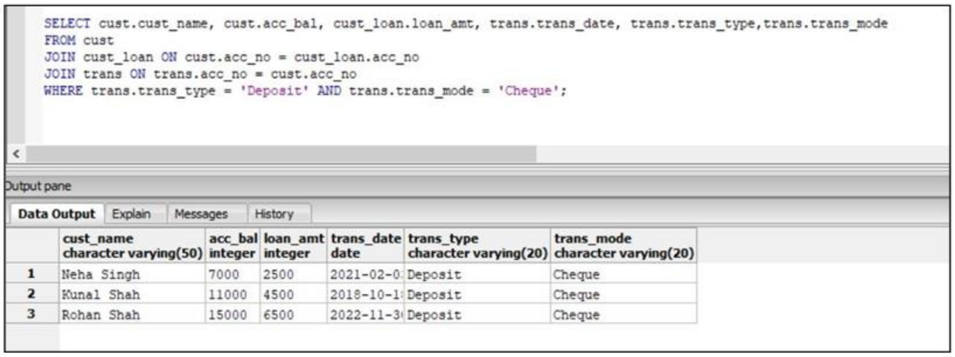
4. Retrieve the details of all transactions made after December 2022



```sql
SELECT b.cust_id, c.cust_name, b.acc_no, t.trans_no, t.trans_type, t.trans_mode, t.trans_amt, t.trans_date
FROM bank_details b
JOIN cust c ON b.cust_id = c.cust_id AND b.acc_no = c.acc_no
JOIN trans t ON b.acc_no = t.acc_no AND b.cust_id = t.cust_id
WHERE t.trans_date > '2022-12-01';
```

	cust_id	cust_name	acc_no	trans_no	trans_type	trans_mode	trans_amt	trans_date
	integer	character varying(50)	integer	integer	character varying(20)	character varying(20)	integer	date
1	13	Amit Kumar	10013	13	Withdrawal	Online	1000	2022-12-1
2	14	Simran Kaur	10014	14	Deposit	NEFT	5000	2022-12-1
3	15	Vikas Mehta	10015	15	Withdrawal	Cash	2000	2022-12-1

5. Retrieve the details of all customers who made deposit through cheque



```sql
SELECT cust.cust_name, cust.acc_bal, cust_loan.loan_amt, trans.trans_date, trans.trans_type,trans.trans_mode
FROM cust
JOIN cust_loan ON cust.acc_no = cust_loan.acc_no
JOIN trans ON trans.acc_no = cust.acc_no
WHERE trans.trans_type = 'Deposit' AND trans.trans_mode = 'Cheque';
```

	cust_name	acc_bal	loan_amt	trans_date	trans_type	trans_mode
	character varying(50)	integer	integer	date	character varying(20)	character varying(20)
1	Neha Singh	7000	2500	2021-02-0	Deposit	Cheque
2	Kunal Shah	11000	4500	2018-10-1	Deposit	Cheque
3	Rohan Shah	15000	6500	2022-11-3	Deposit	Cheque

Q: For the vertical fragments check the correctness rules

Completeness : If relation R is decomposed into fragments R1,R2,....Rn each data item that can be found in R can also be found in one or more Ri's.

Ans: From the above query executions we can say that records present in emp can be found either in cust or in cust_loan.

Reconstruction: If relation R is decomposed into fragments R1,R2,...Rn , it should be possible to define relational operator delta such that R=delta(Ri) for all Ri belongs to Fr.

Ans: When we Join both the tables, we are able to get the original table back, so the reconstruction property stands.

CONCLUSION: Thus, we have performed vertical fragmentation on the database and checked the correctness rules for the same. And after successfully completing this experiment, I learnt that:

1. Vertical fragmentation is a useful technique for improving query performance and data management in sql databases.
2. Vertical fragmentation helps to reduce query processing time.
3. Vertical fragmentation makes easier to update and maintain the data, as well as improve data

security by limiting access to specific columns.

4. Vertical fragmentation can increase the flexibility of the database design