NATIONAL INSTITUTE OF TECHNOLOGY, WARANGAL

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

DBMS PROJECT

DATABASE DESIGN AND IMPLEMENTATION FOR ONLINE PAYMENTS INTERFACE

SUBMITTED BY:

1. **AKSHAT CHOUDHARY 207204**
2. **G. SUNIL KUMAR 207227**
3. **TEJSWO TIWARY 207278**

ACKNOWLEDGEMENT

We would like to express our special thanks of gratitude to Dr.T.Ramakrishnudu, who gave us the golden opportunity to do this wonderful project on database design of an online payments interface , which helped us learn the concepts taught in class and applying them in the real world.

INTRODUCTION TO THE PROBLEM

To design and implement a database for online payments interface, keeping in mind, the various needs and requirements of the users.

DBMS used: Oracle sql developer.

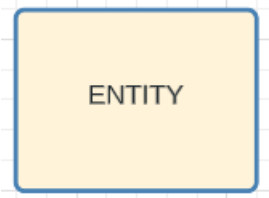
We have modeled the problem into an Entity-relationship model and have described the structure of the database with the help of an ER diagram.

We then converted that to relational model, applying the techniques of normalization so as to reduce the data redundancy in the database.

Entities are any objects, person, class or place with a physical existence in the real world.
In our database, we have considered 7 such entities, each with a set of attributes describing each of those.

1. USER
2. CREDENTIALS
3. ACCOUNT
4. TRANSACTIONS
5. RECEIPT
6. BANK
7. CARD

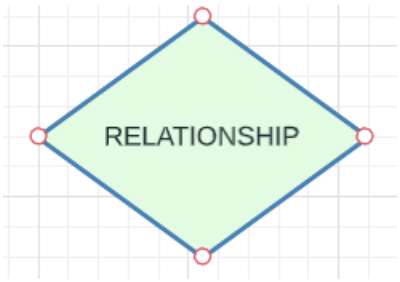
Entities are represented with the help of a rectangle.



Entities have relationships among themselves, the association between entity types. For example, the entites User and Account have a relationship 'HAS' as USER HAS ACCOUNT. Following are the set of relationships used:

1. LOGIN
2. CONTACTS
3. HAS
4. WITH
5. LINKED
6. DEBIT FROM
7. CREDIT TO
8. GENERATES

Relationships are represented with a Rhombus between entities involved in a relationship.



We have assumed only binary relationships during the design of the database.

For a binary relationship, we have marked the cardinality of each of the relationships as:

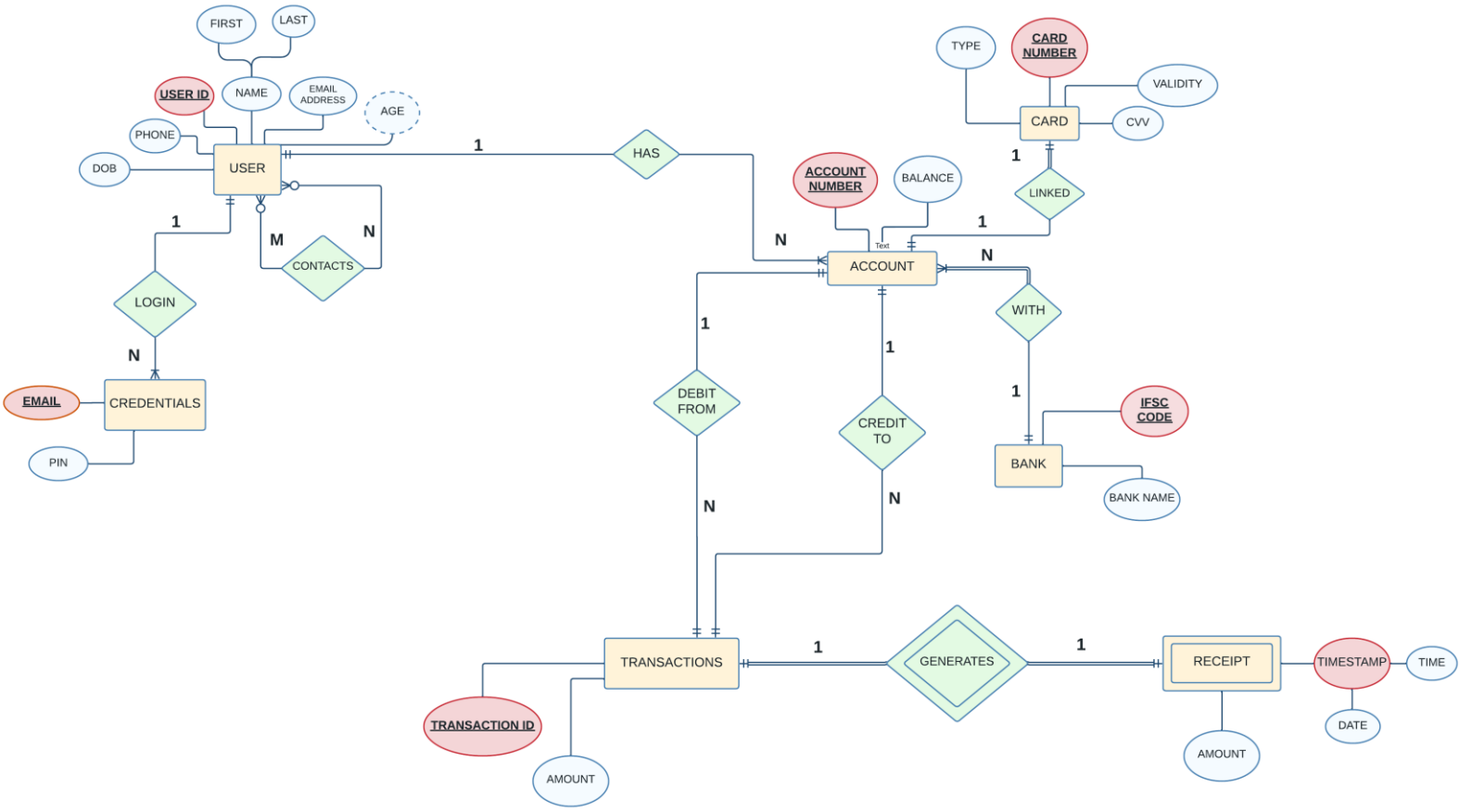
1. 1:1

2. 1:N

3. M:N

‘

ER DIAGRAM:


FIRST
LAST
USER ID
NAME
EMAIL ADDRESS
AGE
PHONE
DOB
USER
1
HAS
TYPE
CARD NUMBER
VALIDITY
CARD
CVV
1
LINKED
ACCOUNT NUMBER
BALANCE
1
N
ACCOUNT
N
WITH
1
M
N
CONTACTS
LOGIN
N
EMAIL
CREDENTIALS
PIN
1
DEBIT FROM
1
CREDIT TO
1
IFSC CODE
BANK
BANK NAME
N
N
TRANSACTIONS
1
GENERATES
1
RECEIPT
TIMESTAMP
TIME
DATE
AMOUNT
TRANSACTION ID
AMOUNT

Here we use the concept of online payment interface to create our own database system.
The above diagram consists of user, credentials, contacts, account, bank, transaction, receipt entities to store data, create shortcut for payment, store history for the previous transaction, and keeping the account information saved in the data. We included some of the information to decide the entities and relationship with cardinality.

ENTITY SET:

1. USER (user_id, first_name, last_name, email_add, dob, phone)

This entity is for to store the personal information of the account holder who performs all the online payment, here USER_ID is the primary key since it will store only one specially allotted number which is not repeated, Then NAME, EMAILADDRESS, PHONE and DOB are normal attributes to store users name, email address, their phone number and their date of birth.

2. CREDENTIALS (email, user_id, pin)

Since UPI or online payment with pin transaction is made to perform transaction with minimal data. USER_ID is primary key to access the accounts linked to this
And another attribute is PIN that stores 4-digit pin or password and its not unique to everyone.

3. ACCOUNT (account_no, balance, user_id, ifsc_code)

This entity is works like a register for the users unique ID where all the records of his/her transaction are kept, 'ACCOUNT_NUMBER' this is primary key since account number is unique to every single individual user, BALANCE which stores amount available at that time in user's account and gets updated after every transaction.

4. TRANSACTIONS (trans_id, amount, credit_to/debit_from)

It stores the information about the transactions. ‘TRANSACTON ID’ this is a primary key since it generates unique number every time after new transaction, next is ‘AMOUNT’ after every transaction amount is recorded.

5. RECEIPT (time_stamp, trans_id, amount)

It is a weak entity set as its existence entirely depends on the transaction entity.
It contains the timestamp of the transaction which is a unique attribute, transaction id is the primary key of the owner entity transaction and attribute amount which stores the transaction amount.

6. BANK (ifsc_code, bank_name)

All the transactions are performed by the bank as user just gives command every bank has its unique ID given as ‘IFSC_CODE’ ifsc code has unique for every bank for example two state bank of India in same area will never have same IFSC code it’s like serial number for banks,‘Bank_name’ is given by the area name where bank is located.

7. CARD (card_no, acc_no, cvv, validity, type_)

Card is for those transaction where UPI is not functionable CARD_NUMBER is primary key (card number is unique on every card that’s why card number is made primary key). ‘CVV’ this attribute is like a lock it’s a 3-4 digit password digit code to procced to payment then next attribute is ‘VALIDITY’ this attribute stores amount of time card has left of service, next attribute is ‘TYPE’ type is for to decide weather card is savings or current

RELATIONSHIP SET:

1. LOGIN
2. CONTACTS
3. HAS
4. WITH
5. LINKED
6. DEBIT FROM

7. CREDIT TO
8. GENERATES

CONVERTING ER DIAGRAM TO RELATIONSHIP SCHEMA:

We converted the ER diagram to relationship schema by first setting up the primary key for each of the entities in the entity sets.

To link two entities that are involved in a relationship, we make use of the Foreign key constraint.

Foreign key has to satisfy the foreign key constraints and are mapped based on the cardinality ratio of relations between the entities in the entity set.

1. **For 1:1 relations**: the primary key of one is taken as foreign key in the other, giving preference to the one that has total participation in the relationship.

2. **For 1:N relations**: include the 1 side's primary key as a foreign key on the N side relation.

3: **for M:N relations**: create a new relation who's primary key is the combination of both entities primary keys and include any relationship attributes.

Thus, we get the relational model of the ER model in the form of a relational schema:

To avoid data modification anomalies, we have applied the concept of normalization.

1NF : Relation contains atomic values.
We have achieved so, by eliminating the multi valued attributes.

2NF: all the non-key attributes are fully functional dependent on the primary key.

RECOGNIZING THE PRIMARY KEY:

By mapping out the functional dependencies of the attributes in each of the relations, we arrived at the primary keys for each of the entities while making sure that the primary key identifies each of the tupple in the relation in a particular relation uniquely.

Below are the primary key constraints taken:

1. USER : **USER_id** -> (user_id, first_name, last_name, email_add, dob, phone)
   Since **USER_id** uniquely identifies all the other attributes, it is chosen as the primary key.

2. CREDENTIALS: **email**-> (email, user_id, pin)
   Since **USER_id** uniquely identifies all the other attributes, it is chosen as the primary key.
   And **user_id** is chosen as the foreign key to reference the user_ entity.
3. ACCOUNT: **ACCOUNT_NO** -> (account_no, balance, user_id, ifsc_code)
   Since **ACCOUNT_NO** uniquely identifies all the other attributes, it is chosen as the primary key.
   user_id is chosen as the foreign key to reference the user_ entity.
   ifsc_code is chosen as the foreign key to reference the bank entity

4. TRANSACTIONS: **TRANSACTION_ID** -> (trans_id, amount, credit_to/debit_from)
   Since **TRANSACTION_ID** uniquely identifies all the other attributes, it is chosen as the primary key.
   credit_to/debit_from are chosen as foreign key to reference the account entity.

5. RECEIPT: **TRANSACTION_ID,TIME_STAMP** -> (time_stamp, trans_id, amount)

Since receipt is a weak entity, the primary key of its owner entity is included as composite key with the partial primary key of its own **(TRANSACTION_ID,TIME_STAMP)** and it **(TRANSACTION_ID)** is also used as foreign key to reference the owner entity.

8. BANK: **IFSC_CODE** -> (ifsc_code, bank_name)

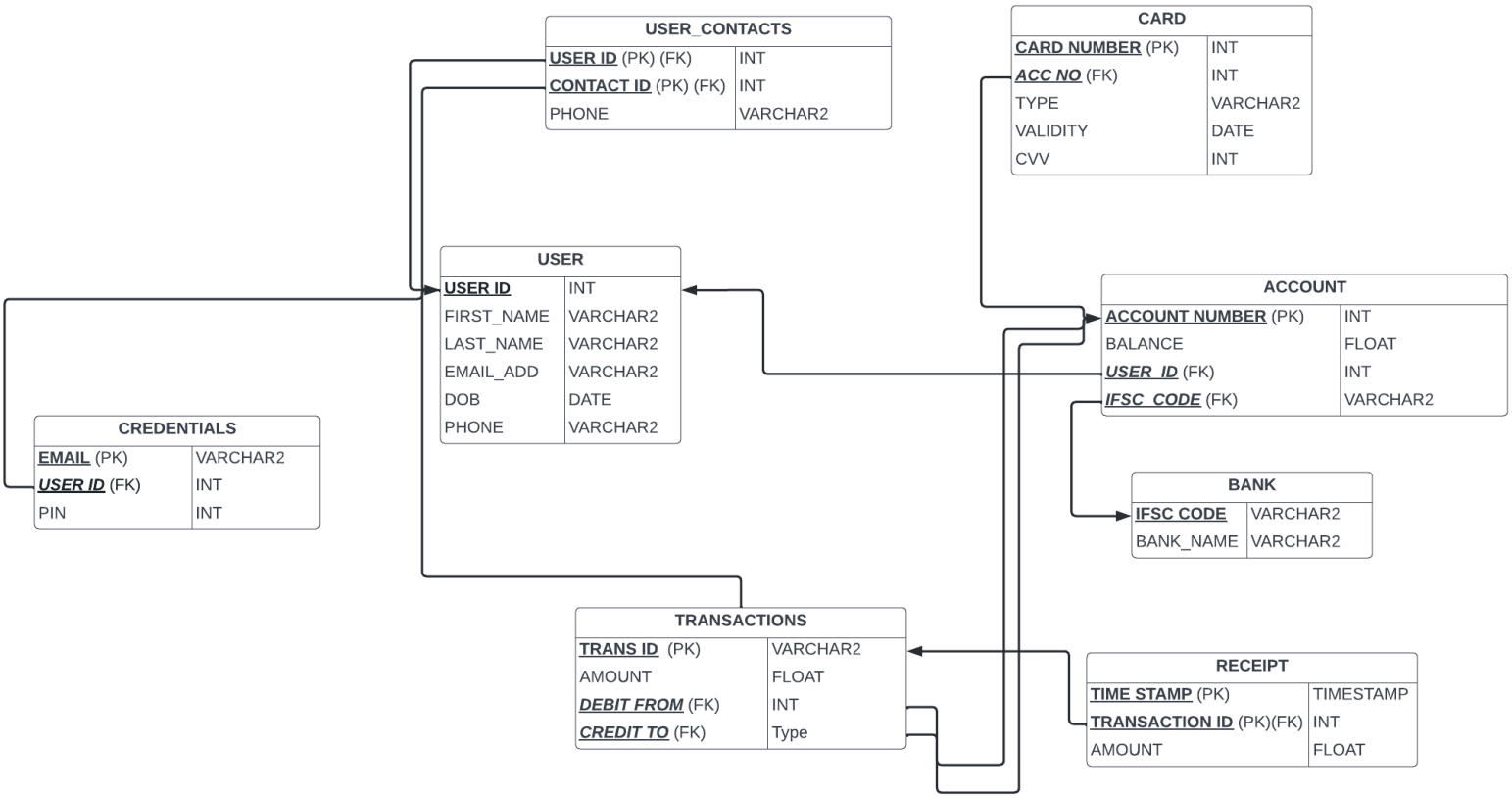
Since **IFSC_CODE** uniquely identifies the other attributes, it is chosen as the primary key.

6. CARD: **CARD_NO** -> (card_no, acc_no, cvv, validity, type_)

Since **CARD_NO** uniquely identifies the other attributes, it is chosen as the primary key.

And acc_no is used to reference the account entity.

RELATIONAL SCHEMA:



CREATION OF TABLES AND INSERTION OF VALUES:

1. User_

```
create table user_ (
  user_id int,
  first_name varchar2 (25),
  last_name varchar2 (25),
  email_add varchar2 (25),
  dob date,
  phone varchar2 (25),
  primary key (user_id)
);
insert into user_ (
  user_id, first_name, last_name, email_add,
  dob, phone
)
values
  (
    100,
    'Gulistanskiy',
    'Zaytseva',
    'Zaytseva@gmail.com',
    to_date('23-jul-2001', 'dd-mm-yyyy'),
    '123456789'
  );
insert into user_ (
  user_id, first_name, last_name, email_add,
  dob, phone
)
values
  (
    101,
    'Daniil',
    'Romanov',
    'Romanov@gmail.com',
    to_date('24-jul-1991', 'dd-mm-yyyy'),
    '1235045489'
  );
insert into user_ (
  user_id, first_name, last_name, email_add,
  dob, phone
)
values
```

```sql
  (
    102,
    'Illarion',
    'Borisov',
    'Borisov@gmail.com',
    to_date('24-aug-1999', 'dd-mm-yyyy'),
    '1546745489'
  );
insert into user_ (
  user_id, first_name, last_name, email_add,
  dob, phone
)
values
  (
    103,
    'Petya',
    'Orlov',
    'Orlov@gmail.com',
    to_date('24-oct-1999', 'dd-mm-yyyy'),
    '1546123489'
  );
insert into user_ (
  user_id, first_name, last_name, email_add,
  dob, phone
)
values
  (
    104,
    'Viktoriya',
    'Sokolova',
    'Sokolova@gmail.com',
    to_date('14-aug-1989', 'dd-mm-yyyy'),
    '1548745489'
  );
insert into user_ (
  user_id, first_name, last_name, email_add,
  dob, phone
)
values
  (
    105,
    'Roza',
    'Koroleva',
    'Koroleva@gmail.com',
    to_date('04-apr-1997', 'dd-mm-yyyy'),
    '1578465489'
  );
```

	USER_ID	FIRST_NAME	LAST_NAME	EMAIL_ADD	DOB	PHONE
1	100	Gulistanskiy	Zaytseva	Zaytseva@gmail.com	23-JUL-01	123456789
2	101	Daniil	Romanov	Romanov@gmail.com	24-JUL-91	1235045489
3	102	Illarion	Borisov	Borisov@gmail.com	24-AUG-99	1546745489
4	103	Petya	Orlov	Orlov@gmail.com	24-OCT-99	1546123489
5	104	Viktoriya	Sokolova	Sokolova@gmail.com	14-AUG-89	1548745489
6	105	Roza	Koroleva	Koroleva@gmail.com	04-APR-97	1578465489

```
Name        Null?    Type
----------- -------- ------------
USER_ID     NOT NULL NUMBER(38)
FIRST_NAME           VARCHAR2(25)
LAST_NAME            VARCHAR2(25)
EMAIL_ADD            VARCHAR2(25)
DOB                  DATE
PHONE                VARCHAR2(25)
```

2. user_contacts:

```
create table user_contacts (
  user_id int,
  contact_id int,
  phone varchar2 (25),
  primary key (user_id, contact_id),
  foreign key (user_id) references user_ (user_id),
  foreign key (contact_id) references user_ (user_id)
);
insert into user_contacts (user_id, contact_id, phone)
values
  (100, 101, '1235045489');
insert into user_contacts (user_id, contact_id, phone)
values
  (100, 102, '1546745489');
insert into user_contacts (user_id, contact_id, phone)
values
  (101, 100, '123456789');
insert into user_contacts (user_id, contact_id, phone)
values
  (102, 105, '1578465489');
insert into user_contacts (user_id, contact_id, phone)
```

```
values
  (102, 103, '1546123489');
insert into user_contacts (user_id, contact_id, phone)
values
  (103, 104, '1548745489');
insert into user_contacts (user_id, contact_id, phone)
values
  (104, 102, '1546745489');
insert into user_contacts (user_id, contact_id, phone)
values
  (105, 102, '1546745489');
```

	USER_ID	CONTACT_ID	PHONE
1	100	101	1235045489
2	100	102	1546745489
3	101	100	123456789
4	102	105	1578465489
5	102	103	1546123489
6	103	104	1548745489
7	104	102	1546745489

```
Name        Null?    Type
---------- -------- ------------
USER_ID    NOT NULL NUMBER(38)
CONTACT_ID NOT NULL NUMBER(38)
PHONE               VARCHAR2(25)
```

3. credentials

```
create table credentials (
  email varchar2 (25),
  user_id int,
  pin number (4),
  primary key (user_id),
  foreign key (user_id) references user_(user_id)
);
insert into credentials (email, user_id, pin)
values
  ('Zaytseva@gmail.com', 100, 3432);
insert into credentials (email, user_id, pin)
values
  ('Romanov@gmail.com', 101, 3765);
insert into credentials (email, user_id, pin)
values
  ('Borisov@gmail.com', 102, 3012);
insert into credentials (email, user_id, pin)
values
  ('Orlov@gmail.com', 103, 9432);
insert into credentials (email, user_id, pin)
values
  ('Sokolova@gmail.com', 104, 3442);
```

```sql
insert into credentials (email, user_id, pin)
values
  ('Koroleva@gmail.com', 105, 3333);
```

	EMAIL	USER_ID	PIN
1	Zaytseva@gmail.com	100	3432
2	Romanov@gmail.com	101	3765
3	Borisov@gmail.com	102	3012
4	Orlov@gmail.com	103	9432
5	Sokolova@gmail.com	104	3442
6	Koroleva@gmail.com	105	3333

```
Name     Null?    Type
------- -------- ------------
EMAIL             VARCHAR2(25)
USER_ID NOT NULL  NUMBER(38)
PIN               NUMBER(4)
```

4. bank

```sql
create table bank (
  ifsc_code varchar2 (25),
  bank_name varchar2 (25),
  primary key (ifsc_code)
);
insert into bank (ifsc_code, bank_name)
values
  (
    'SBIN0013258', 'State Bank of India'
  );
insert into bank (ifsc_code, bank_name)
values
  (
    'SBIN0031466', 'State Bank of India '
  );
insert into bank (ifsc_code, bank_name)
values
  (
    'PUNB0046800', 'Punjab National Bank'
  );
insert into bank (ifsc_code, bank_name)
values
  (
    'UBIN0554430', 'Union Bank of India'
  );
insert into bank (ifsc_code, bank_name)
values
  ('UTIB0001032', 'Axis bank');
insert into bank (ifsc_code, bank_name)
values
```

```
  ('ICIC0000915', 'ICICI Bank');
```

	IFSC_CODE	BANK_NAME
1	SBIN0013258	State Bank of India
2	SBIN0031466	State Bank of India
3	PUNB0046800	Punjab National Bank
4	UBIN0554430	Union Bank of India
5	UTIB0001032	Axis bank
6	ICIC0000915	ICICI Bank

```
Name      Null?    Type
--------- -------- ------------
IFSC_CODE NOT NULL VARCHAR2(25)
BANK_NAME          VARCHAR2(25)
```

5.Account

```
create table account_ (
  account_no int,
  balance float,
  user_id int,
  ifsc_code varchar2 (25),
  primary key (account_no),
  foreign key (user_id) references user_ (user_id),
  foreign key (ifsc_code) references bank (ifsc_code)
);
insert into account_ (
  account_no, balance, user_id, ifsc_code
)
values
  (
    1746101, 15000.96, 100, 'ICIC0000915'
  );
insert into account_ (
  account_no, balance, user_id, ifsc_code
)
values
  (
    1746102, 15070.96, 101, 'UTIB0001032'
  );
insert into account_ (
  account_no, balance, user_id, ifsc_code
)
values
  (
    2123546, 15230.96, 102, 'SBIN0013258'
  );
insert into account_ (
```

```
  account_no, balance, user_id, ifsc_code
)
values
  (
    1010123, 15230.96, 103, 'SBIN0031466'
  );
insert into account_ (
  account_no, balance, user_id, ifsc_code
)
values
  (
    5254325, 3424.36, 102, 'PUNB0046800'
  );
insert into account_ (
  account_no, balance, user_id, ifsc_code
)
values
  (
    5328725, 15880.96, 102, 'SBIN0013258'
  );
```

	ACCOUNT_NO	BALANCE	USER_ID	IFSC_CODE
1	1746101	15000.96	100	ICIC0000915
2	1746102	15070.96	101	UTIB0001032
3	2123546	15230.96	102	SBIN0013258
4	1010123	15230.96	103	SBIN0031466
5	5254325	3424.36	102	PUNB0046800
6	5328725	15880.96	102	SBIN0013258

```
Name        Null?    Type
----------- -------- ------------
ACCOUNT_NO  NOT NULL NUMBER(38)
BALANCE              FLOAT(126)
USER_ID              NUMBER(38)
IFSC_CODE            VARCHAR2(25)
```

6. Card

```
create table card (
  card_no varchar2 (12),
  acc_no int,
  type_ varchar2 (25),
  validity date,
  cvv number (3),
  primary key (card_no),
  foreign key (acc_no) references account_ (account_no)
);
insert into card (
  card_no, acc_no, cvv, validity, type_
)
```

```sql
values
  (
    '432156121234',
    1746101,
    137,
    to_date ('23-feb-2023', 'dd-mm-yyyy'),
    'SAVINGS'
  );
insert into card (
  card_no, acc_no, cvv, validity, type_
)
values
  (
    '432156548234',
    1746101,
    147,
    to_date ('23-feb-2023', 'dd-mm-yyyy'),
    'SAVINGS'
  );
insert into card (
  card_no, acc_no, cvv, validity, type_
)
values
  (
    '533361950372',
    2123546,
    066,
    to_date ('25-dec-2025', 'dd-mm-yyyy'),
    'CURRENT'
  );
insert into card (
  card_no, acc_no, cvv, validity, type_
)
values
  (
    '922458533215',
    1010123,
    234,
    to_date ('01-dec-2022', 'dd-mm-yyyy'),
    'CURRENT'
  );
insert into card (
  card_no, acc_no, cvv, validity, type_
)
values
  (
    '879655456894',
    5254325,
    874,
```

```
    to_date ('31-mar-2025', 'dd-mm-yyyy'),
    'SAVINGS'
  );
insert into card (
  card_no, acc_no, cvv, validity, type_
)
values
  (
    '365415789687',
    5328725,
    545,
    to_date ('11-may-2026', 'dd-mm-yyyy'),
    'CURRENT'
  );
```

	CARD_NO	ACC_NO	TYPE_	VALIDITY	CVV
1	432156121234	1746101	SAVINGS	23-FEB-23	137
2	533361950372	2123546	CURRENT	25-DEC-25	66
3	922458533215	1010123	CURRENT	01-DEC-22	234
4	879655456894	5254325	SAVINGS	31-MAR-25	874
5	365415789687	5328725	CURRENT	11-MAY-26	545
6	432156548234	1746101	SAVINGS	23-FEB-23	147

```
Name      Null?    Type
--------- -------- ------------
CARD_NO   NOT NULL VARCHAR2(12)
ACC_NO             NUMBER(38)
TYPE_              VARCHAR2(25)
VALIDITY           DATE
CVV                NUMBER(3)
```

7. Transactions

```
create table transactions (
  trans_id varchar2 (25),
  amount float,
  debit_from int,
  credit_to int,
  primary key (trans_id),
  foreign key (debit_from) references account_ (account_no),
  foreign key (credit_to) references account_ (account_no)
);
insert into transactions (trans_id, amount, debit_from)
values
  ('12DD4569', 2335, 1746101);
insert into transactions (trans_id, amount, credit_to)
values
```

```
  ('45AA2003', 4122, 1746101);
insert into transactions (trans_id, amount, debit_from)
values
  ('24SM2021', 9855, 1010123);
insert into transactions (trans_id, amount, debit_from)
values
  ('08MM2008', 7588, 5328725);
insert into transactions (trans_id, amount, debit_from)
values
  ('06JJ2013', 6933, 5254325);
insert into transactions (trans_id, amount, debit_from)
values
  ('05MM2015', 2154, 1010123);
insert into transactions (trans_id, amount, credit_to)
values
  ('06SS2020', 7485, 1746101);
insert into transactions (trans_id, amount, credit_to)
values
  ('12SS2020', 3562, 1010123);
insert into transactions (trans_id, amount, credit_to)
values
  ('09DC2020', 9123, 5328725);
```

	TRANS_ID	AMOUNT	DEBIT_FROM	CREDIT_TO
1	12DD4569	2335	1746101	(null)
2	45AA2003	4122	(null)	1746101
3	24SM2021	9855	1010123	(null)
4	08MM2008	7588	5328725	(null)
5	06JJ2013	6933	5254325	(null)
6	05MM2015	2154	1010123	(null)
7	06SS2020	7485	(null)	1746101
8	12SS2020	3562	(null)	1010123
9	09DC2020	9123	(null)	5328725

```
Name        Null?    Type
---------- -------- ------------
TRANS_ID    NOT NULL VARCHAR2(25)
AMOUNT               FLOAT(126)
DEBIT_FROM           NUMBER(38)
CREDIT_TO            NUMBER(38)
```

8. Receipt

```
create table receipt (
  time_stamp timestamp,
  trans_id varchar2 (25),
  amount float,
  primary key (time_stamp, trans_id),
  foreign key (trans_id) references transactions (trans_id)
);
insert into receipt (time_stamp, trans_id, amount)
values
  (
```

```sql
    to_timestamp (
      '2022-01-02:04:05', 'YYYY-MM-DD HH24:MI:SS'
    ),
    '12DD4569',
    2335
  );
insert into receipt (time_stamp, trans_id, amount)
values
  (
    to_timestamp (
      '2022-01-02:04:55', 'YYYY-MM-DD HH24:MI:SS'
    ),
    '45AA2003',
    4122
  );
insert into receipt (time_stamp, trans_id, amount)
values
  (
    to_timestamp (
      '2022-01-03:04:05', 'YYYY-MM-DD HH24:MI:SS'
    ),
    '24SM2021',
    9855
  );
insert into receipt (time_stamp, trans_id, amount)
values
  (
    to_timestamp (
      '2022-01-03:04:05', 'YYYY-MM-DD HH24:MI:SS'
    ),
    '08MM2008',
    7588
  );
insert into receipt (time_stamp, trans_id, amount)
values
  (
    to_timestamp (
      '2022-01-20:04:05', 'YYYY-MM-DD HH24:MI:SS'
    ),
    '06JJ2013',
    6933
  );
insert into receipt (time_stamp, trans_id, amount)
values
  (
    to_timestamp (
      '2022-01-03:04:55', 'YYYY-MM-DD HH24:MI:SS'
    ),
    '05MM2015',
```

```
    2154
  );
insert into receipt (time_stamp, trans_id, amount)
values
  (
    to_timestamp (
      '2024-01-02:23:05', 'YYYY-MM-DD HH24:MI:SS'
    ),
    '06SS2020',
    7485
  );
insert into receipt (time_stamp, trans_id, amount)
values
  (
    to_timestamp (
      '2023-01-03:23:05', 'YYYY-MM-DD HH24:MI:SS'
    ),
    '12SS2020',
    3562
  );
insert into receipt (time_stamp, trans_id, amount)
values
  (
    to_timestamp (
      '2022-01-20:04:05', 'YYYY-MM-DD HH24:MI:SS'
    ),
    '09DC2020',
    9123
  );
```

	EMAIL	USER_ID	PIN
1	Zaytseva@gmail.com	100	3432
2	Romanov@gmail.com	101	3765
3	Borisov@gmail.com	102	3012
4	Orlov@gmail.com	103	9432
5	Sokolova@gmail.com	104	3442
6	Koroleva@gmail.com	105	3333

```
Name     Null?    Type
------- -------- ------------
EMAIL             VARCHAR2(25)
USER_ID NOT NULL  NUMBER(38)
PIN               NUMBER(4)
```

TRIGGERS:

1. checkDate

```
create
or replace trigger checkDate before insert on card
for each row declare cur_date date;
begin cur_date := sysdate;
if : new.validity < cur_date then
raise_application_error (-20001, 'invalid date');
end if;
end;
```

2. checkDOB

```
 create or replace trigger checkDOB
before insert on user_ for each row
declare
cur_date date;
begin
cur_date := sysdate;
if :new.dob > cur_date
then
raise_application_error (-20001,'invalid date');
end if;
end;
```

Sample QUERIES:

1. List out all the transactions where amount > 5000.

```
select * from transactions where amount>5000;
```

cript Output × Query Result ×

SQL | All Rows Fetched: 5 in 0.005 seconds

	TRANS_ID	AMOUNT	DEBIT_FROM	CREDIT_TO
1	24SM2021	9855	1010123	(null)
2	08MM2008	7588	5328725	(null)
3	06JJ2013	6933	5254325	(null)
4	06SS2020	7485	(null)	1746101
5	09DC2020	9123	(null)	5328725

2. List out all the different bank names the users have account with.

```
select  distinct bank_name from bank ;
```

ript Output × Query Result ×

SQL | All Rows Fetched: 6 in 0.006 seconds

	BANK_NAME
1	Punjab National Bank
2	Axis bank
3	State Bank of India
4	Union Bank of India
5	ICICI Bank

3. Print ifsc, bank name of account number 1010123.

```
select  distinct bank.ifsc_code, bank.bank_name from
account_, bank
where bank.ifsc_code = ( select account_.ifsc_code from account_ where
account_.account_no = 1010123 );
```

Script Output × Query Result ×

SQL | All Rows Fetched: 1 in 0.008 seconds

	IFSC_CODE	BANK_NAME
1	SBIN0031466	State Bank of India

4. List out all the debit transactions, ifsc_code performed from account 1010123.

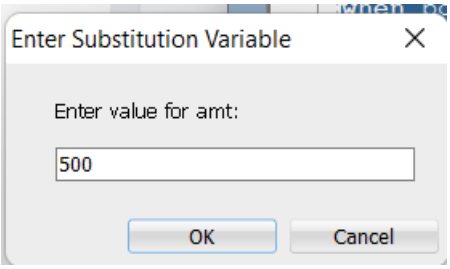
```
select  transactions.trans_id, transactions.amount,
transactions.debit_from,account_.ifsc_code from transactions,account_
where transactions.debit_from=1010123 and  account_.account_no =1010123 ;
```

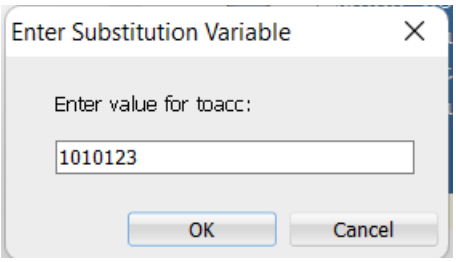
ript Output × | Query Result ×

SQL | All Rows Fetched: 2 in 0.004 seconds

	TRANS_ID	AMOUNT	DEBIT_FROM	IFSC_CODE
1	24SM2021	9855	1010123	SBIN0031466
2	05MM2015	2154	1010123	SBIN0031466

5. Perform a transaction.





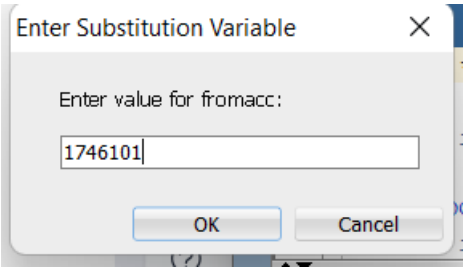


Table before:

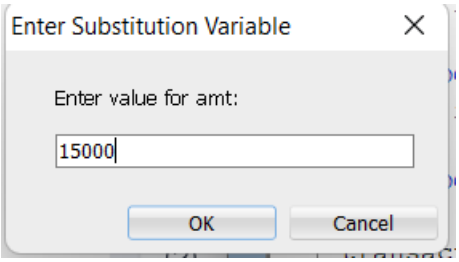
	CARD_NO	ACC_NO	TYPE_	VALIDITY	CVV
1	432156121234	1746101	SAVINGS	23-FEB-23	137
2	533361950372	2123546	CURRENT	25-DEC-25	66
3	922458533215	1010123	CURRENT	01-DEC-22	234
4	879655456894	5254325	SAVINGS	31-MAR-25	874
5	365415789687	5328725	CURRENT	11-MAY-26	545
6	432156548234	1746101	SAVINGS	23-FEB-23	147

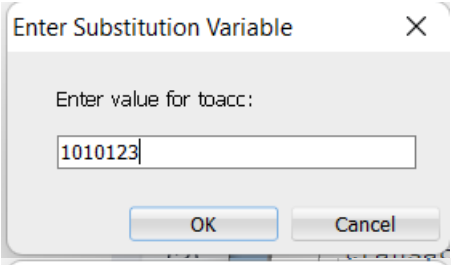
Table after:

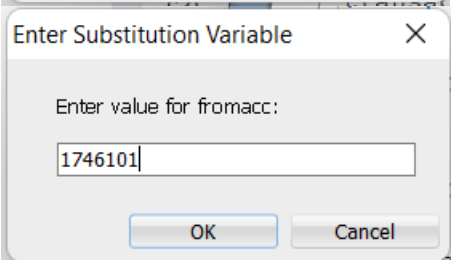
	ACCOUNT_NO	BALANCE	USER_ID	IFSC_CODE
1	1746101	14500.96	100	ICIC0000915
2	1746102	15070.96	101	UTIB0001032
3	2123546	15230.96	102	SBIN0013258
4	1010123	15730.96	103	SBIN0031466
5	5254325	3424.36	102	PUNB0046800
6	5328725	15880.96	102	SBIN0013258

```
declare
amt float;
toacc int;
fromacc int;
bal float;
bounce_account exception;
begin
amt:= &amt;
toacc:= &toacc;
fromacc:= &fromacc;
select balance into bal from
account_ where fromacc = account_no;
if ( amt > bal ) then
raise bounce_account;
else
update account_ set
balance = (balance - amt) where account_no = fromacc;
update account_ set
balance = (balance + amt) where account_no = toacc;

dbms_output.put_line ('transaction successful!!');
end if;
exception
when bounce_account then
dbms_output.put_line ('insufficient balance!!');
when others then
dbms_output.put_line ('invalid!!');
end;
/
```




Enter Substitution Variable
Enter value for toacc:
1010123
OK
Cancel


Enter Substitution Variable
Enter value for fromacc:
1746101
OK
Cancel