19BIT0179	Saksham Arora
19BIT0156	Akshat Mishra
19BIT0164	Swetank Kaushik

D A T A B A S E
M A N A G E M E N T
S Y S T E M S

Library Management System

ABSTRACT

(UoD)

The mini world we are going to use for the implementation of our ‘Library Management’ database system is somewhat based on/inter-related to our own VIT college for example. The Library management database system has been made for a campus, in which the students of a supposed college or higher institution studies. The Library needs some form of management due to the presence of not only so many books and thesaurus, but also due to the large number of users (be it students or faculty) that throughout the years will spend their time to study and sometimes borrow and return the books. There can be many other factors that require management as well, like the students not returning the books on time would then have to give a fine, which may be different for all the users. Hence there is a need for proper organization of various types of information/data, as most of the duty of which falls on the shoulders of the Library administrators who are responsible for keeping records of various types of data of staff, faculty, books (and all their details), students and their respective fine (if any). All of this can be facilitated by the help of the Library management database system, which will not only enable students to keep track of their own Library-related data, but allow a means through which the library managers can carry out their responsibilities with the maximum efficiency and obtain desired results with ease.

DATA REQUIREMENTS

Entities:

1) **User** is an entity type which has attributes like **Username**, Name, Password, and ‘Is Staff?’. **Username** is the key attribute. Every person using the Library’s Online Portal registers himself with a username and password.

2) **Student/Faculty** which is a subclass of User which has attributes Date-of-Birth, Gender, Email, Address, Penalty, ‘Is Debarred?’ and Department. Which specifies the user if he/she’s a student or a faculty member.

3) **Staff** is also a subclass of User which has attributes like id, salary and job type.

4) **Book** is an entity type which has attributes like **ISBN**, Cost, Edition, Title, Publisher and Author(s). **ISBN** is the key attribute here. Author is a Multi-Valued attribute.

5) **Book Copy** which is a weak entity type having it’s identifying relationship as “Has” and owner entity being **Book**. Its attributes are Copy#, “Is on hold?”, “Is checked out?”, “Is damaged?” and Future Requester. Book Copy helps us to understand which copy# of the book goes to a particular user and how many copies of a particular book are still available in the library.

6) **Subject** has its attributes as **Name**, and Keyword(s) where keyword(s) is a multi-valued attribute. Name is the key attribute. **Name** is the key attribute. Every book has a Subject, on which it is written.

7) **Shelf** is another entity type having attributes as **shelf#** and aisle# which indicates on which shelf and aisle a particular book is kept. **Shelf#** is the key attribute.

8) **Floor** is an entity type having attributes as **Floor#**, Number of student assistant and Number of Copiers. Floor helps the user to navigate to the book and is related to shelf also. **Floor#** is the key attribute.

Relationships:

1) <u>Student/Faculty **issues** a Book Copy</u> (M:N)

   "Many" users can take "Any" number of books available. Every Issue has its own issue_id, date of issue, extension date and return date. A Student or a faculty may or may not issue a book. Therefore, there is a partial participation of the entity type **Student/Faculty** towards the relationship **issues**.

2) <u>Every Book **has** a Book copy</u> (1:N)

   A book can have any number of copies. **Has** is the identifying relationship for the weak entity – Book copy. Every book copy has a book on which is based; so, there's total participation of **book copy** towards the relationship **has**.

3) <u>A book **belongs to** a subject</u> (N:1)

   A Subject can have any number of books. A book has to belong to at least one of the subjects; so, **book** has total participation towards the relationship **belongs to.**

4) <u>A subject **is kept on** floor</u> (N:1)

   A floor can have any number of subjects. A subject must be kept in at least one of the floors; so, there's total participation of **subject** towards the relationship **is kept on.**

5) <u>Every Floor **accommodates** a shelf</u> (1:N)

   A floor can have any number of shelves. A shelf must be kept in at least one of the floors; so, there's total participation of **shelf** towards the relationship **accommodates.**

6) <u>A Book may be **located on** a shelf</u> (N:1)

   A shelf can have any number of books. At any time, a book can be with a user or it can be placed on a shelf. So, there's not a total participation of Book towards **located on.**

FUNCTIONAL REQUIREMENTS

1) RETRIEVAL OF DATA:

i) Search:

To get the search results of the books, and their details like-book name, author, ISBN number, and the number of copies left.

ii) Track Location:

To search on which floor/shelf/aisle, the book is kept

iii) Extension:

If the user wants to extend the date, the last date for submitting the book back to the library gets extended and the current date, old date and the extended date is visible with the number of days left.

iv) Popular subject:

The admin will be able to look at what subjects are popular for a particular month, and number of checkouts for the particular subject.

v) Frequent user:

The admin gets the names of frequent users on the basis of maximum number of checkouts.

vi) Popular book:

Admin will get details of the popular book, on the basis of the number of checkouts of the books' copy.

vii) Damaged book report:

Admin will be able to see which books are damaged, and what all books need to be reordered/repaired.

viii) Fine details:

The user will get the information regarding the late submission and hence the fine that is to be paid (which is set by the admin).

2) REMOVAL OF OLD DATA

i) **Deletion of post final year accounts:**

When students pass out, after completing their studies, their names should get removed from the database.

ii) **Book copy lost:**

When a book copy gets lost, it gets deleted from the database and the total count of book copies is decreased.

iii) **Staff id removal:**

If a faculty resigns/retires then their staff-id gets removed from the database.

iv) **Blacklisted students:**

If a student breaks the norms of the library, his/her account will get banned/deleted.

3) DATA UPDATION:

i) **Checkout book:**

Whenever a book is checked out of the library, the database of the library gets updated.

ii) **Future hold:**

If a book is not available currently in the library, then the user has the option of requesting a future hold, so that whenever the book gets available, the requested user will get an email from the library.

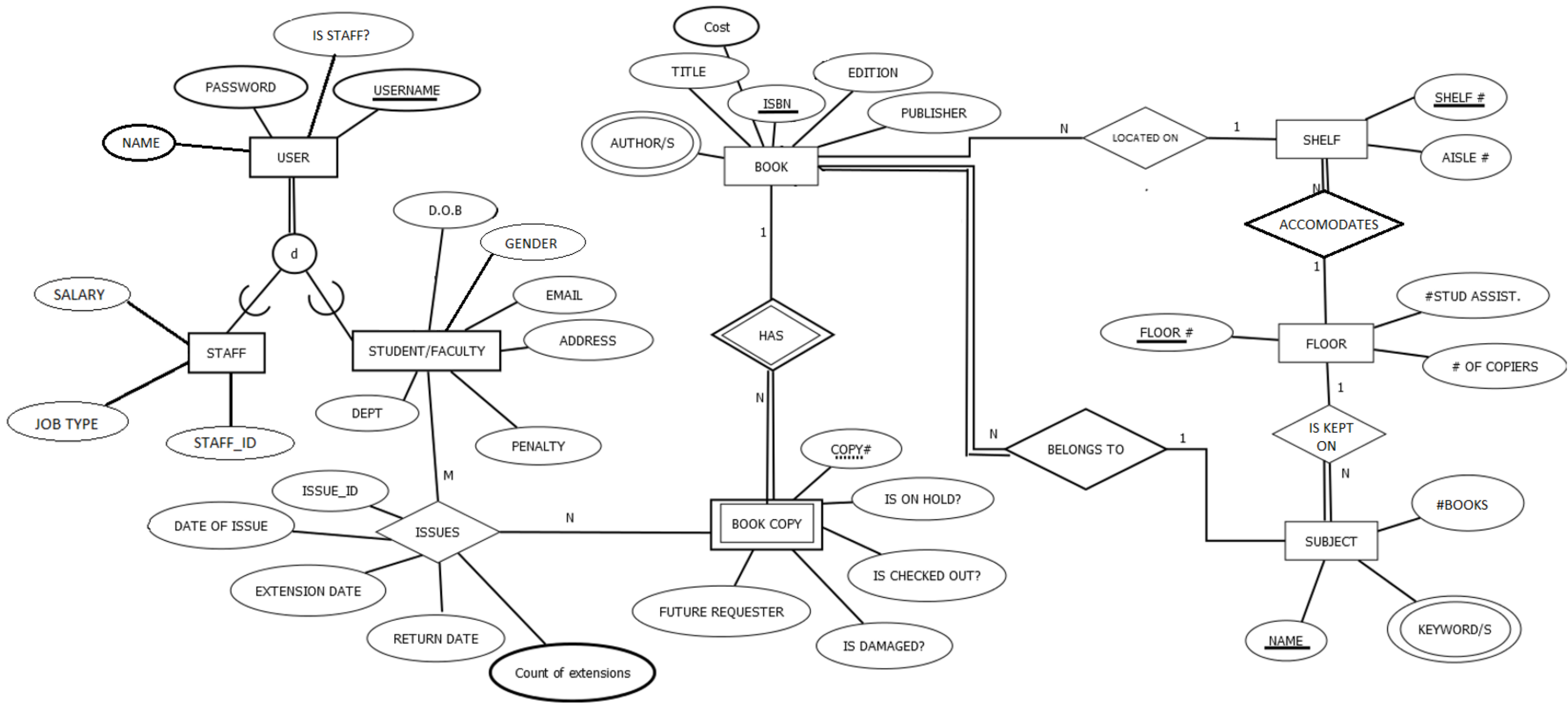
iii) **Return:**

When a user returns a book in the library, the database will be updated.

iv) **User Creation:**

Whenever a new user registers herself/himself, data updation is required.

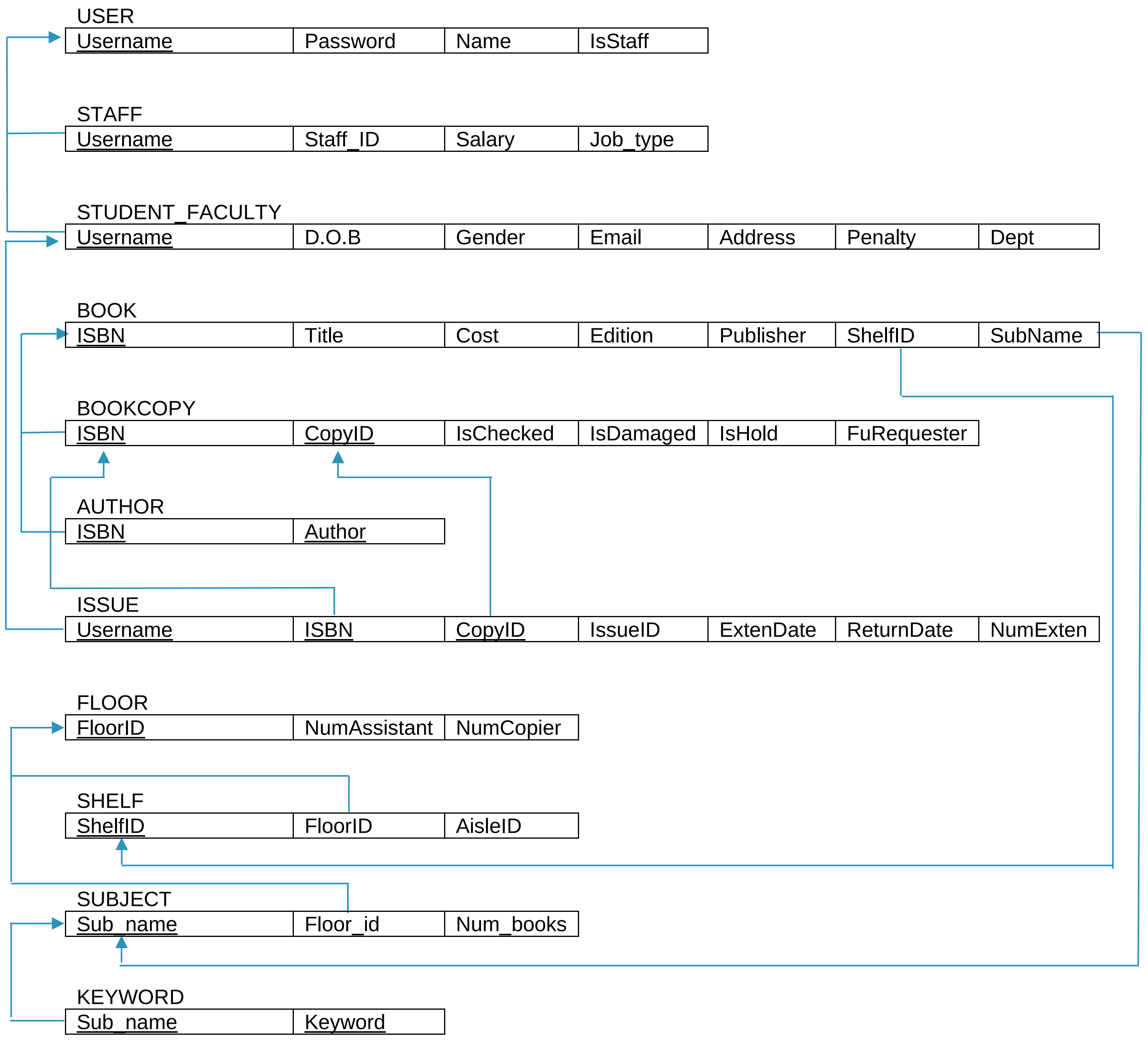
ER diagram:



IS STAFF?
PASSWORD
USERNAME
NAME
USER
d
SALARY
STAFF
JOB TYPE
STAFF_ID
D.O.B
GENDER
EMAIL
ADDRESS
STUDENT/FACULTY
DEPT
PENALTY
M
ISSUE_ID
DATE OF ISSUE
ISSUES
EXTENSION DATE
RETURN DATE
Count of extensions
N
Cost
TITLE
ISBN
EDITION
PUBLISHER
AUTHOR/S
BOOK
1
HAS
N
COPY#
BOOK COPY
IS ON HOLD?
IS CHECKED OUT?
FUTURE REQUESTER
IS DAMAGED?
N
LOCATED ON
1
SHELF
SHELF #
AISLE #
N
ACCOMODATES
1
#STUD ASSIST.
FLOOR #
FLOOR
# OF COPIERS
1
IS KEPT ON
N
N
BELONGS TO
1
SUBJECT
#BOOKS
NAME
KEYWORD/S

REVIEW – 2

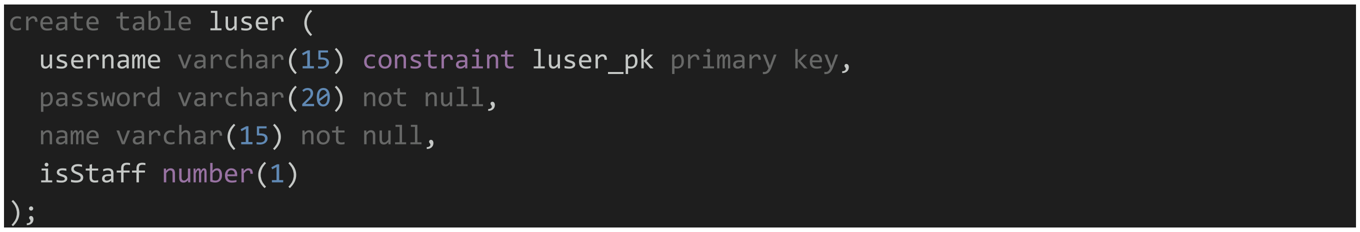
Relational schema diagram:


USER
Username | Password | Name | IsStaff
STAFF
Username | Staff_ID | Salary | Job_type
STUDENT_FACULTY
Username | D.O.B | Gender | Email | Address | Penalty | Dept
BOOK
ISBN | Title | Cost | Edition | Publisher | ShelfID | SubName
BOOKCOPY
ISBN | CopyID | IsChecked | IsDamaged | IsHold | FuRequester
AUTHOR
ISBN | Author
ISSUE
Username | ISBN | CopyID | IssueID | ExtenDate | ReturnDate | NumExten
FLOOR
FloorID | NumAssistant | NumCopier
SHELF
ShelfID | FloorID | AisleID
SUBJECT
Sub_name | Floor_id | Num_books
KEYWORD
Sub_name | Keyword

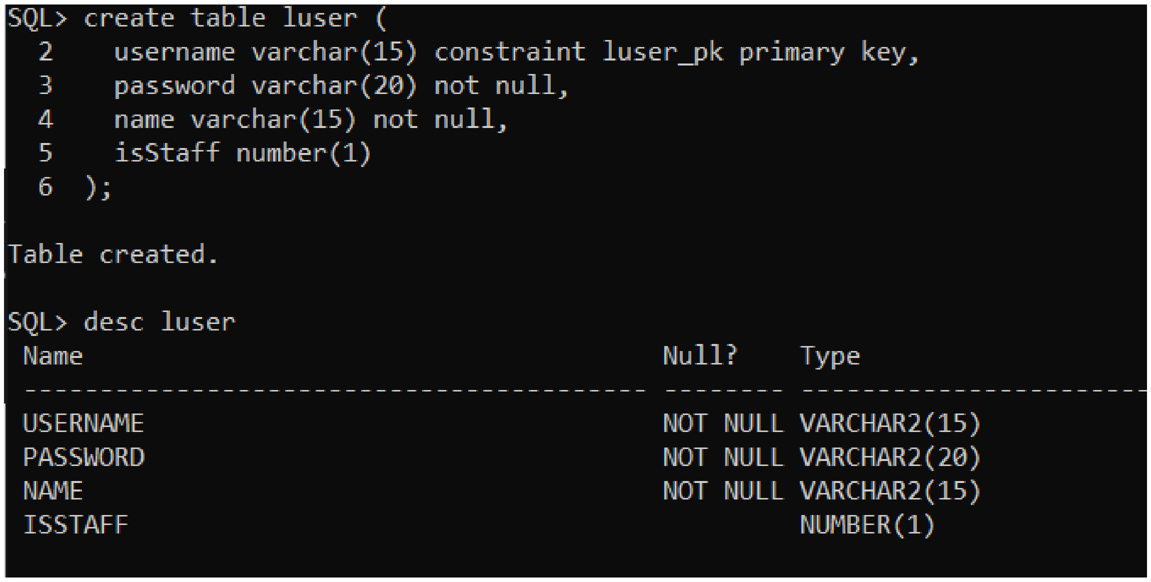
Tables:

- USER
- STAFF
- STUDENT_FACULTY
- BOOK
- BOOKCOPY
- AUTHOR
- ISSUE
- FLOOR
- SHELF
- SUBJECT
- KEYWORD

USER



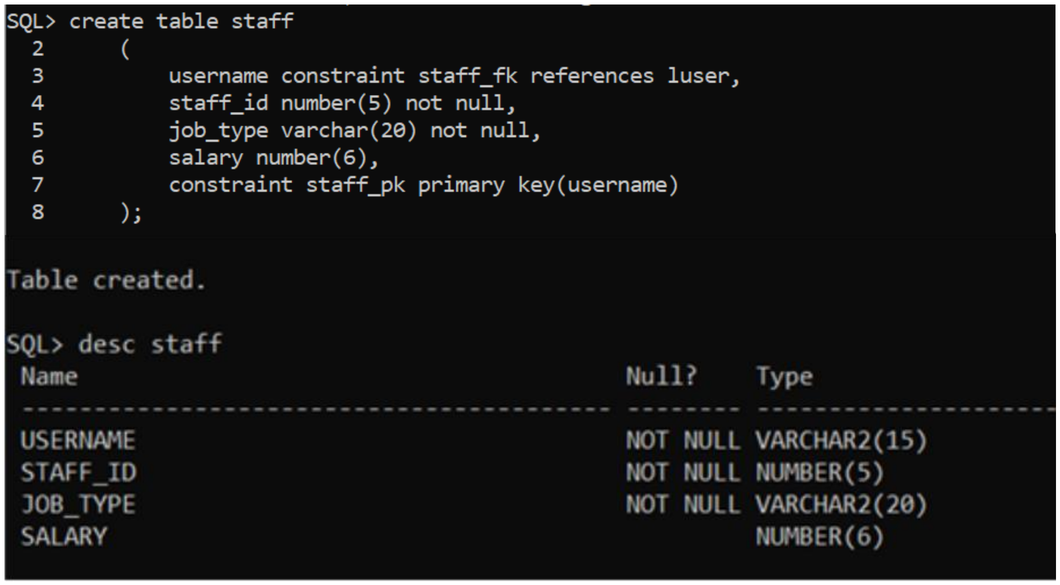
```
create table luser (
  username varchar(15) constraint luser_pk primary key,
  password varchar(20) not null,
  name varchar(15) not null,
  isStaff number(1)
);
```



```
SQL> create table luser (
  2    username varchar(15) constraint luser_pk primary key,
  3    password varchar(20) not null,
  4    name varchar(15) not null,
  5    isStaff number(1)
  6  );

Table created.

SQL> desc luser
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 USERNAME                                  NOT NULL VARCHAR2(15)
 PASSWORD                                  NOT NULL VARCHAR2(20)
 NAME                                      NOT NULL VARCHAR2(15)
 ISSTAFF                                            NUMBER(1)
```

STAFF

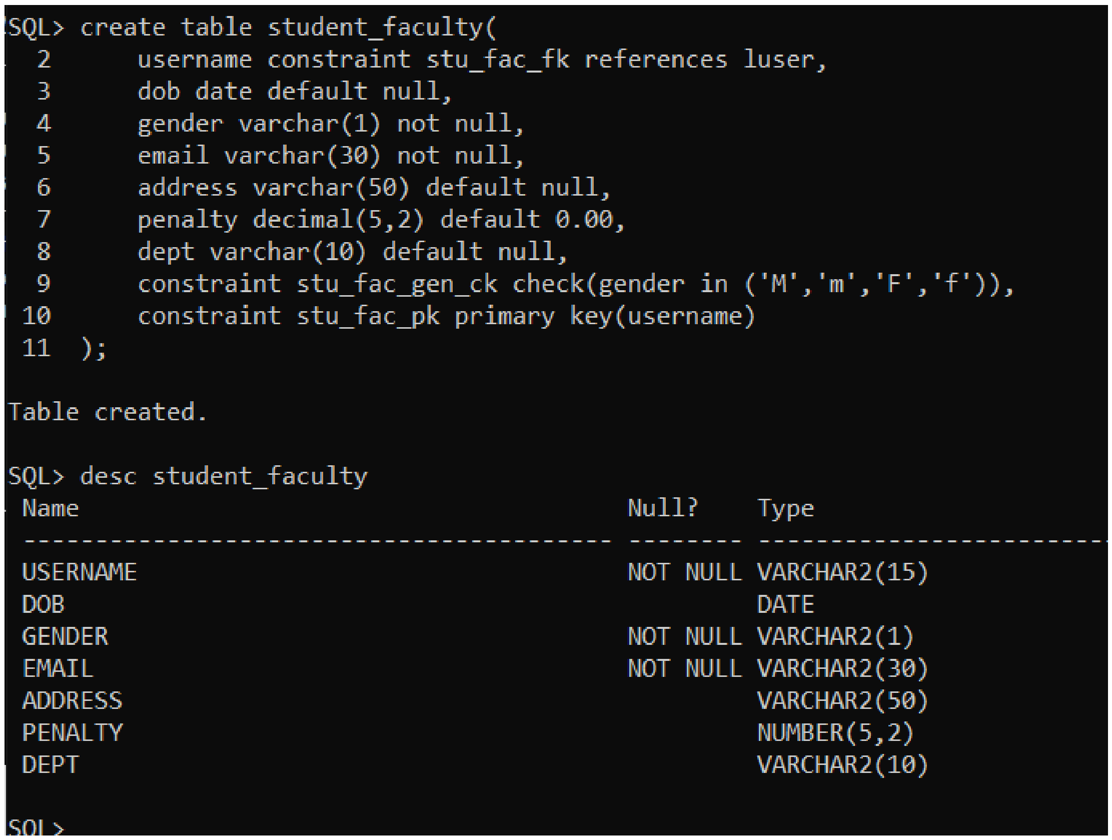
```
create table staff (
  username constraint staff_fk references luser,
  staff_id number(5) not null,
  job_type varchar(20) not null,
  salary number(6),
  constraint staff_pk primary key(username)
);
```



```
SQL> create table staff
  2     (
  3         username constraint staff_fk references luser,
  4         staff_id number(5) not null,
  5         job_type varchar(20) not null,
  6         salary number(6),
  7         constraint staff_pk primary key(username)
  8     );

Table created.

SQL> desc staff
 Name                                      Null?    Type
 ----------------------------------------- -------- --------------------
 USERNAME                                  NOT NULL VARCHAR2(15)
 STAFF_ID                                  NOT NULL NUMBER(5)
 JOB_TYPE                                  NOT NULL VARCHAR2(20)
 SALARY                                             NUMBER(6)
```

STUDENT_FACULTY

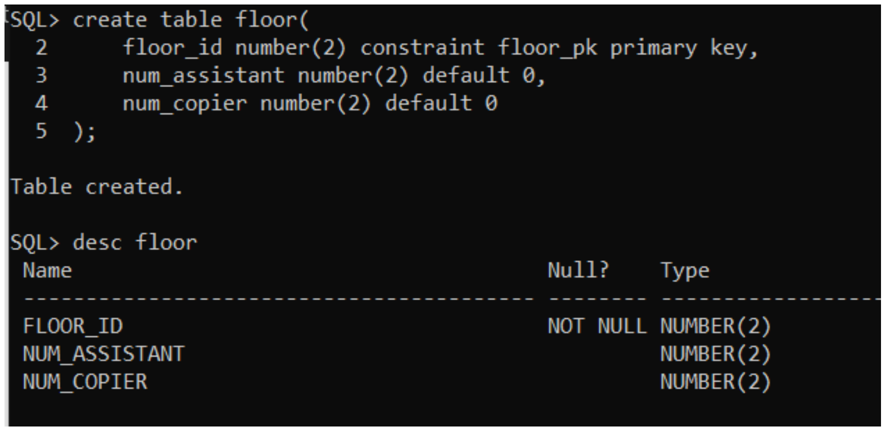
```
create table student_faculty(
    username constraint stu_fac_fk references luser,
    dob date default null,
    gender varchar(1) not null,
    email varchar(30) not null,
    address varchar(50) default null,
    penalty decimal(5,2) default 0.00,
    dept varchar(10) default null,
    constraint stu_fac_gen_ck check(gender in ('M','m','F','f')),
    constraint stu_fac_pk primary key(username)
);
```



```
SQL> create table student_faculty(
  2      username constraint stu_fac_fk references luser,
  3      dob date default null,
  4      gender varchar(1) not null,
  5      email varchar(30) not null,
  6      address varchar(50) default null,
  7      penalty decimal(5,2) default 0.00,
  8      dept varchar(10) default null,
  9      constraint stu_fac_gen_ck check(gender in ('M','m','F','f')),
 10      constraint stu_fac_pk primary key(username)
 11  );

Table created.

SQL> desc student_faculty
 Name                                      Null?    Type
 ----------------------------------------- -------- ------------------------
 USERNAME                                  NOT NULL VARCHAR2(15)
 DOB                                                DATE
 GENDER                                    NOT NULL VARCHAR2(1)
 EMAIL                                     NOT NULL VARCHAR2(30)
 ADDRESS                                            VARCHAR2(50)
 PENALTY                                            NUMBER(5,2)
 DEPT                                               VARCHAR2(10)

SQL>
```

FLOOR

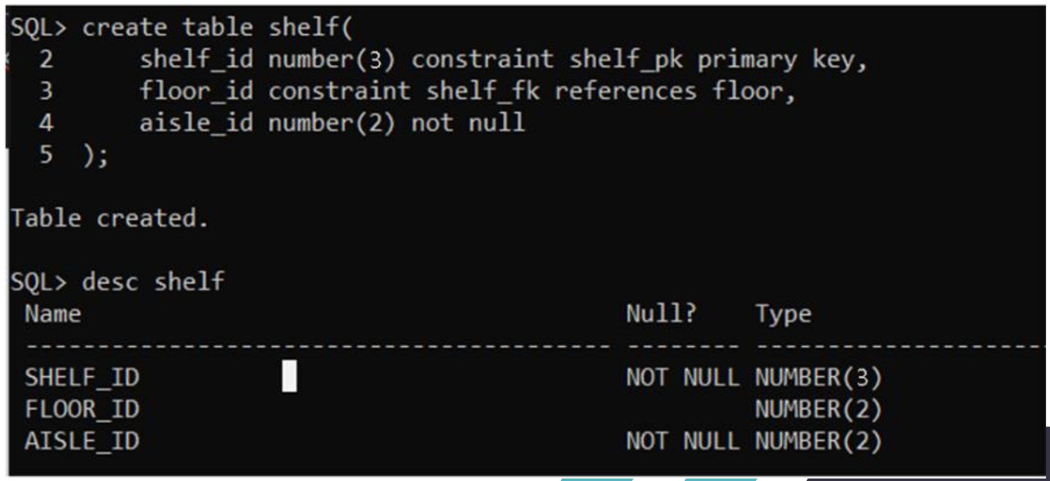
```
create table floor(
    floor_id number(2) constraint floor_pk primary key,
    num_assistant number(2) default 0,
    num_copier number(2) default 0
);
```



```
SQL> create table floor(
  2      floor_id number(2) constraint floor_pk primary key,
  3      num_assistant number(2) default 0,
  4      num_copier number(2) default 0
  5  );

Table created.

SQL> desc floor
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------
 FLOOR_ID                                  NOT NULL NUMBER(2)
 NUM_ASSISTANT                                      NUMBER(2)
 NUM_COPIER                                         NUMBER(2)
```

SHELF

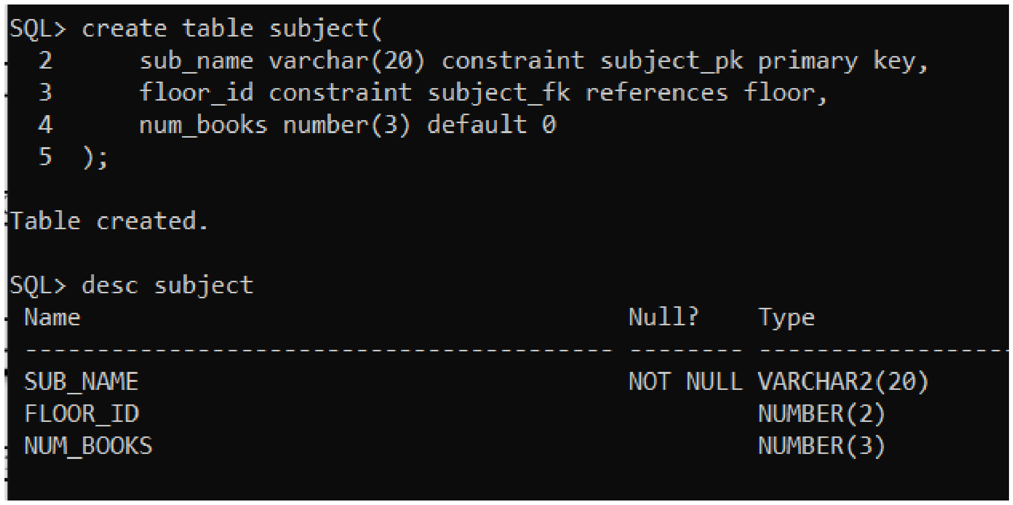
```
create table shelf(
    shelf_id number(3) constraint shelf_pk primary key,
    floor_id constraint shelf_fk references floor,
    aisle_id number(2) not null
);
```



```
SQL> create table shelf(
  2      shelf_id number(3) constraint shelf_pk primary key,
  3      floor_id constraint shelf_fk references floor,
  4      aisle_id number(2) not null
  5  );

Table created.

SQL> desc shelf
 Name                                      Null?    Type
 ----------------------------------------- -------- --------------------
 SHELF_ID                                  NOT NULL NUMBER(3)
 FLOOR_ID                                           NUMBER(2)
 AISLE_ID                                  NOT NULL NUMBER(2)
```

SUBJECT

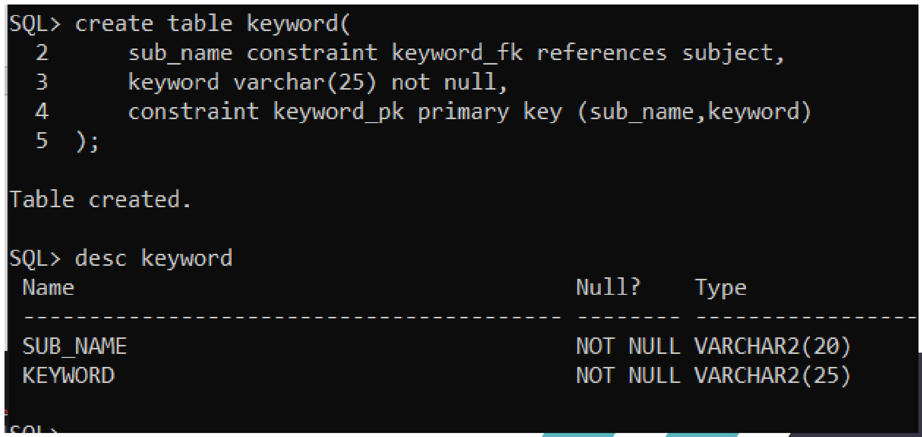
```
create table subject(
    sub_name varchar(20) constraint subject_pk primary key,
    floor_id constraint subject_fk references floor,
    num_books number(3) default 0
);
```



```
SQL> create table subject(
  2      sub_name varchar(20) constraint subject_pk primary key,
  3      floor_id constraint subject_fk references floor,
  4      num_books number(3) default 0
  5  );

Table created.

SQL> desc subject
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------
 SUB_NAME                                  NOT NULL VARCHAR2(20)
 FLOOR_ID                                           NUMBER(2)
 NUM_BOOKS                                          NUMBER(3)
```

KEYWORD

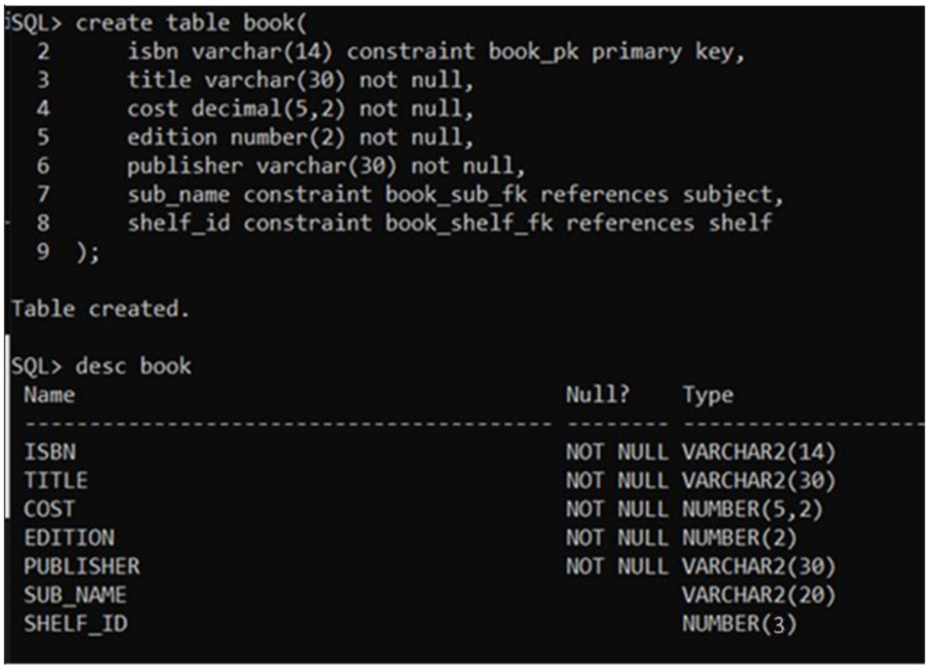
```
create table keyword(
    sub_name constraint keyword_fk references subject,
    keyword varchar(25) not null,
    constraint keyword_pk primary key (sub_name,keyword)
);
```



```
SQL> create table keyword(
  2      sub_name constraint keyword_fk references subject,
  3      keyword varchar(25) not null,
  4      constraint keyword_pk primary key (sub_name,keyword)
  5  );

Table created.

SQL> desc keyword
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------
 SUB_NAME                                  NOT NULL VARCHAR2(20)
 KEYWORD                                   NOT NULL VARCHAR2(25)

SQL>
```

BOOK

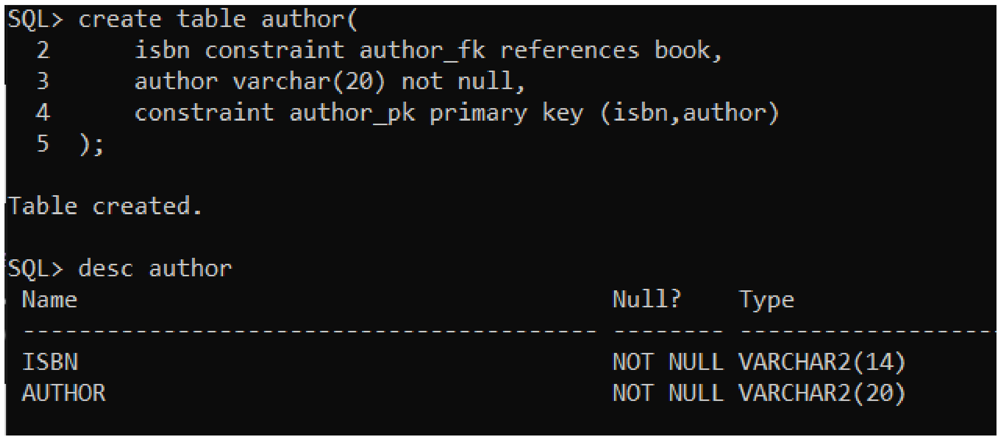
```
create table book(
    isbn varchar(14) constraint book_pk primary key,
    title varchar(30) not null,
    cost decimal(5,2) not null,
    edition number(2) not null,
    publisher varchar(30) not null,
    sub_name constraint book_sub_fk references subject,
    shelf_id constraint book_shelf_fk references shelf
);
```



```
SQL> create table book(
  2      isbn varchar(14) constraint book_pk primary key,
  3      title varchar(30) not null,
  4      cost decimal(5,2) not null,
  5      edition number(2) not null,
  6      publisher varchar(30) not null,
  7      sub_name constraint book_sub_fk references subject,
  8      shelf_id constraint book_shelf_fk references shelf
  9  );

Table created.

SQL> desc book
 Name                                      Null?    Type
 ----------------------------------------- -------- ------------------
 ISBN                                      NOT NULL VARCHAR2(14)
 TITLE                                     NOT NULL VARCHAR2(30)
 COST                                      NOT NULL NUMBER(5,2)
 EDITION                                   NOT NULL NUMBER(2)
 PUBLISHER                                 NOT NULL VARCHAR2(30)
 SUB_NAME                                           VARCHAR2(20)
 SHELF_ID                                           NUMBER(3)
```

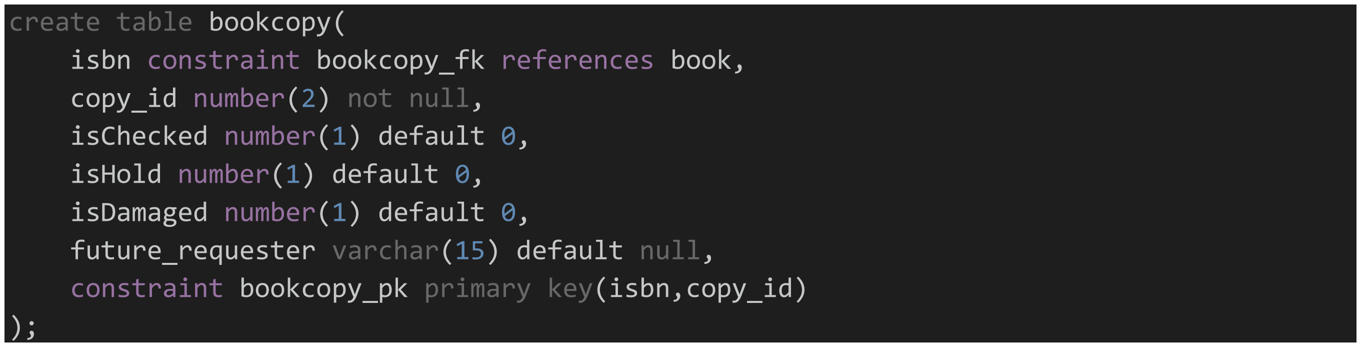
AUTHOR

```
create table author(
    isbn constraint author_fk references book,
    author varchar(20) not null,
    constraint author_pk primary key (isbn,author)
);
```

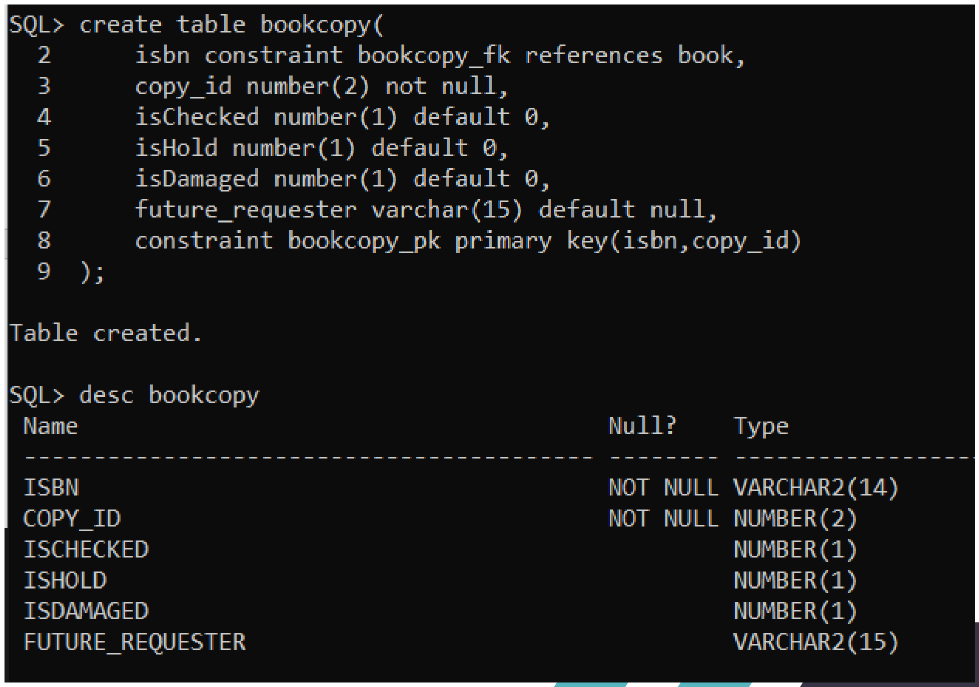


```
SQL> create table author(
  2      isbn constraint author_fk references book,
  3      author varchar(20) not null,
  4      constraint author_pk primary key (isbn,author)
  5  );

Table created.

SQL> desc author
 Name                                      Null?    Type
 ----------------------------------------- -------- -------------------
 ISBN                                      NOT NULL VARCHAR2(14)
 AUTHOR                                    NOT NULL VARCHAR2(20)
```

BOOKCOPY

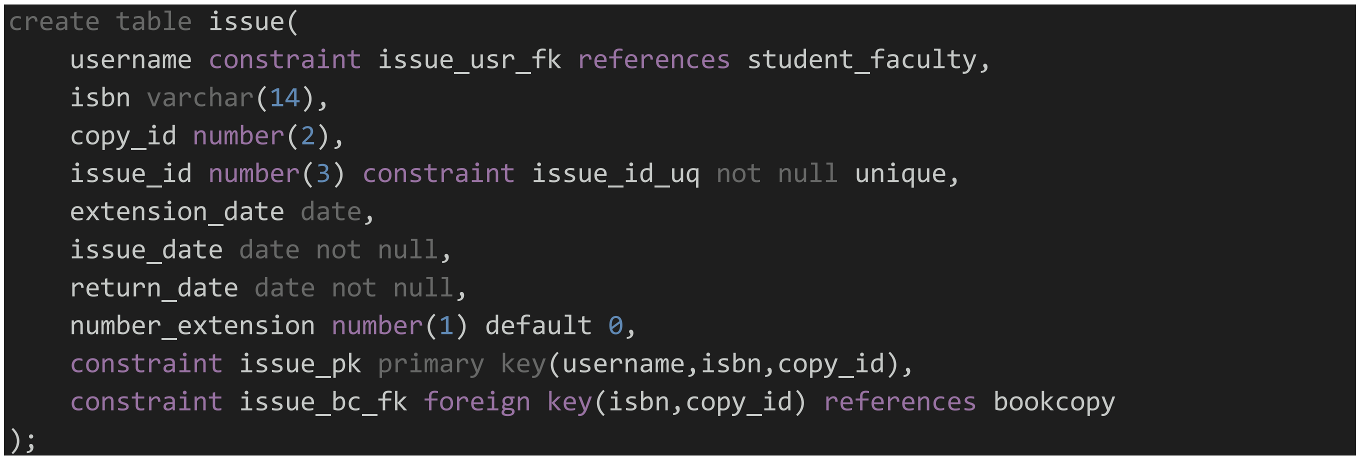


```
create table bookcopy(
    isbn constraint bookcopy_fk references book,
    copy_id number(2) not null,
    isChecked number(1) default 0,
    isHold number(1) default 0,
    isDamaged number(1) default 0,
    future_requester varchar(15) default null,
    constraint bookcopy_pk primary key(isbn,copy_id)
);
```

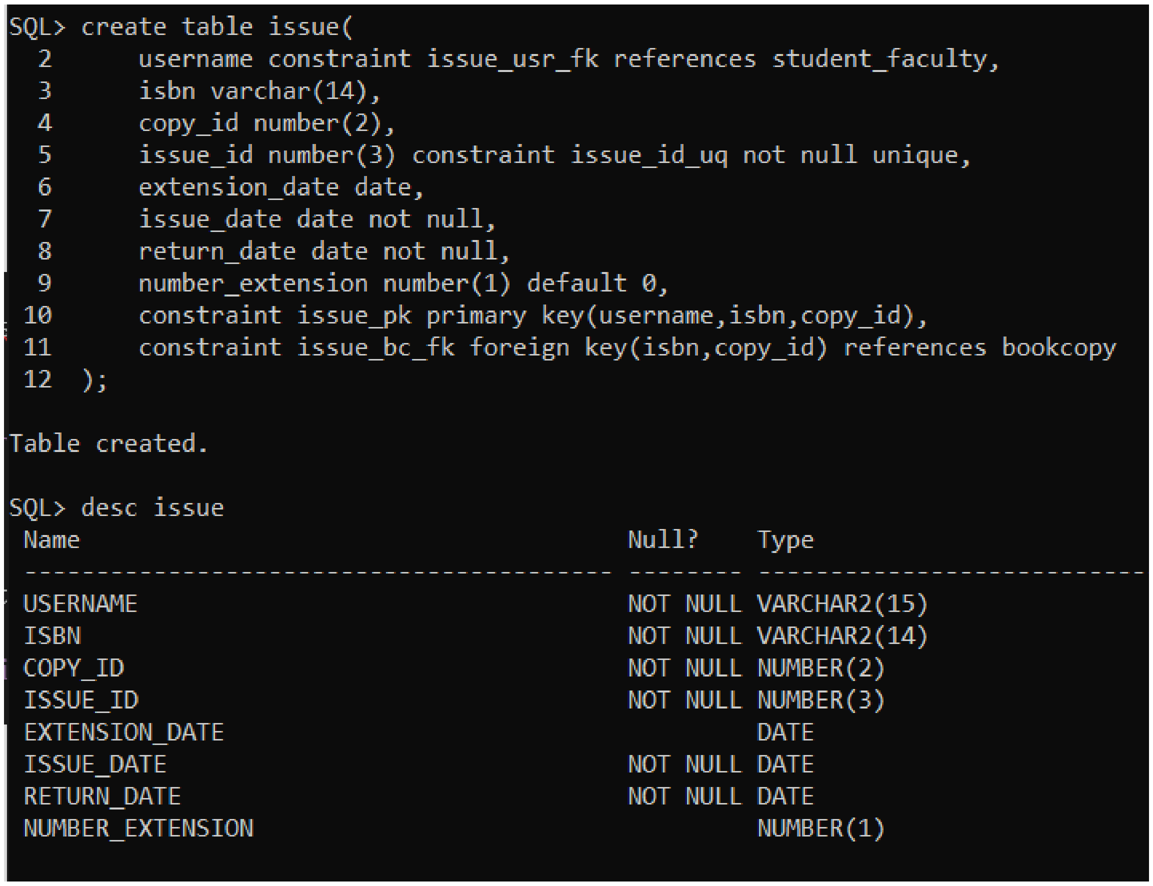


```
SQL> create table bookcopy(
  2      isbn constraint bookcopy_fk references book,
  3      copy_id number(2) not null,
  4      isChecked number(1) default 0,
  5      isHold number(1) default 0,
  6      isDamaged number(1) default 0,
  7      future_requester varchar(15) default null,
  8      constraint bookcopy_pk primary key(isbn,copy_id)
  9  );

Table created.

SQL> desc bookcopy
 Name                                      Null?    Type
 ----------------------------------------- -------- -------------------
 ISBN                                      NOT NULL VARCHAR2(14)
 COPY_ID                                   NOT NULL NUMBER(2)
 ISCHECKED                                          NUMBER(1)
 ISHOLD                                             NUMBER(1)
 ISDAMAGED                                          NUMBER(1)
 FUTURE_REQUESTER                                   VARCHAR2(15)
```

ISSUE



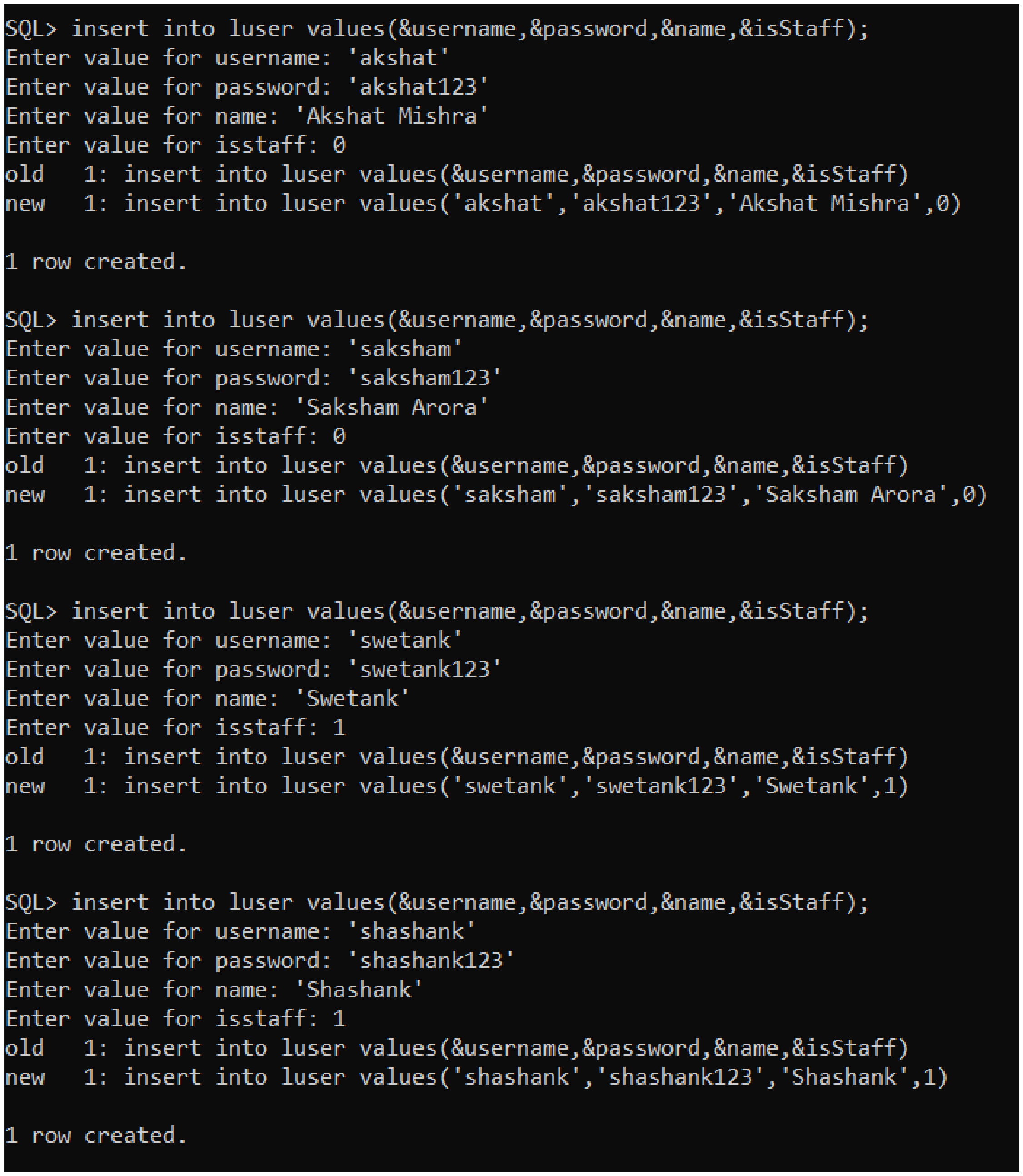
```
create table issue(
    username constraint issue_usr_fk references student_faculty,
    isbn varchar(14),
    copy_id number(2),
    issue_id number(3) constraint issue_id_uq not null unique,
    extension_date date,
    issue_date date not null,
    return_date date not null,
    number_extension number(1) default 0,
    constraint issue_pk primary key(username,isbn,copy_id),
    constraint issue_bc_fk foreign key(isbn,copy_id) references bookcopy
);
```



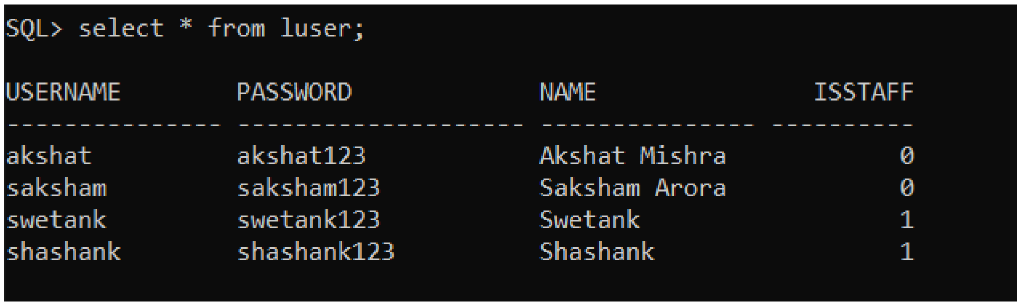
```
SQL> create table issue(
        username constraint issue_usr_fk references student_faculty,
        isbn varchar(14),
        copy_id number(2),
        issue_id number(3) constraint issue_id_uq not null unique,
        extension_date date,
        issue_date date not null,
        return_date date not null,
        number_extension number(1) default 0,
        constraint issue_pk primary key(username,isbn,copy_id),
        constraint issue_bc_fk foreign key(isbn,copy_id) references bookcopy
     );

Table created.

SQL> desc issue
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 USERNAME                                  NOT NULL VARCHAR2(15)
 ISBN                                      NOT NULL VARCHAR2(14)
 COPY_ID                                   NOT NULL NUMBER(2)
 ISSUE_ID                                  NOT NULL NUMBER(3)
 EXTENSION_DATE                                     DATE
 ISSUE_DATE                                NOT NULL DATE
 RETURN_DATE                               NOT NULL DATE
 NUMBER_EXTENSION                                   NUMBER(1)
```

Inserting values

USER

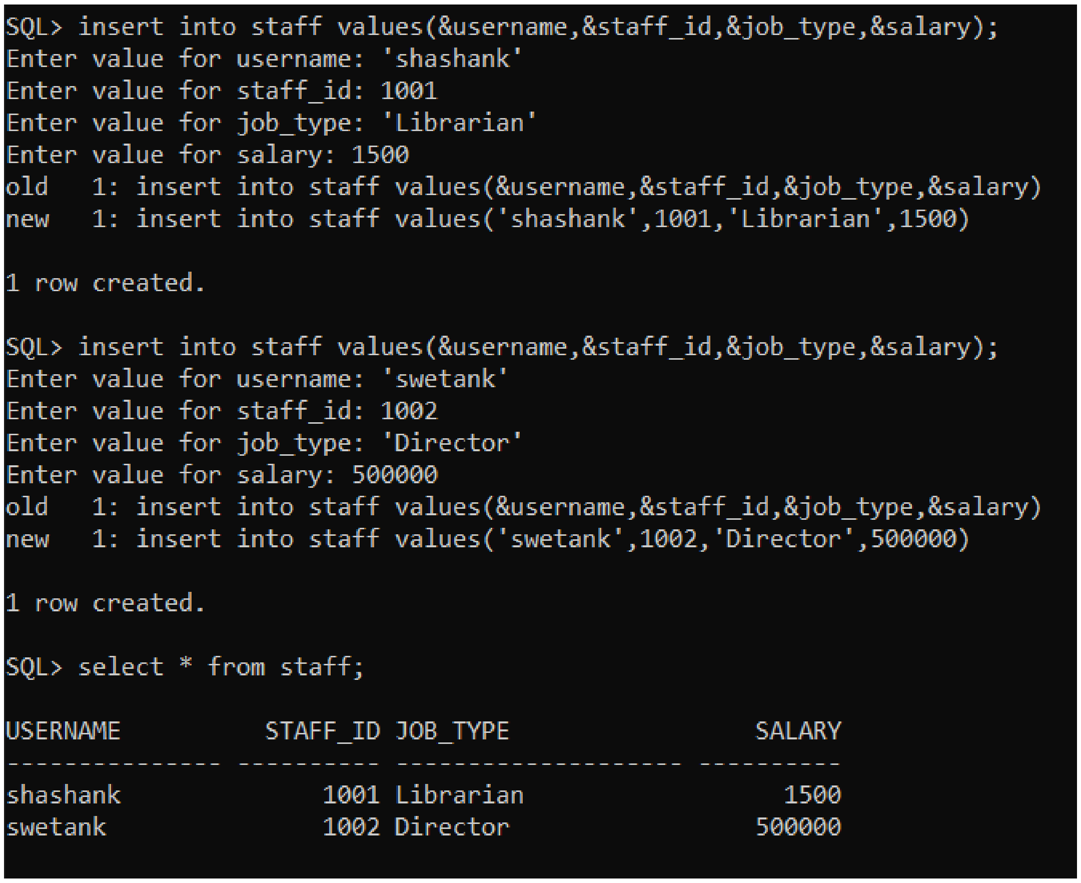


```
SQL> insert into luser values(&username,&password,&name,&isStaff);
Enter value for username: 'akshat'
Enter value for password: 'akshat123'
Enter value for name: 'Akshat Mishra'
Enter value for isstaff: 0
old   1: insert into luser values(&username,&password,&name,&isStaff)
new   1: insert into luser values('akshat','akshat123','Akshat Mishra',0)

1 row created.

SQL> insert into luser values(&username,&password,&name,&isStaff);
Enter value for username: 'saksham'
Enter value for password: 'saksham123'
Enter value for name: 'Saksham Arora'
Enter value for isstaff: 0
old   1: insert into luser values(&username,&password,&name,&isStaff)
new   1: insert into luser values('saksham','saksham123','Saksham Arora',0)

1 row created.

SQL> insert into luser values(&username,&password,&name,&isStaff);
Enter value for username: 'swetank'
Enter value for password: 'swetank123'
Enter value for name: 'Swetank'
Enter value for isstaff: 1
old   1: insert into luser values(&username,&password,&name,&isStaff)
new   1: insert into luser values('swetank','swetank123','Swetank',1)

1 row created.

SQL> insert into luser values(&username,&password,&name,&isStaff);
Enter value for username: 'shashank'
Enter value for password: 'shashank123'
Enter value for name: 'Shashank'
Enter value for isstaff: 1
old   1: insert into luser values(&username,&password,&name,&isStaff)
new   1: insert into luser values('shashank','shashank123','Shashank',1)

1 row created.
```



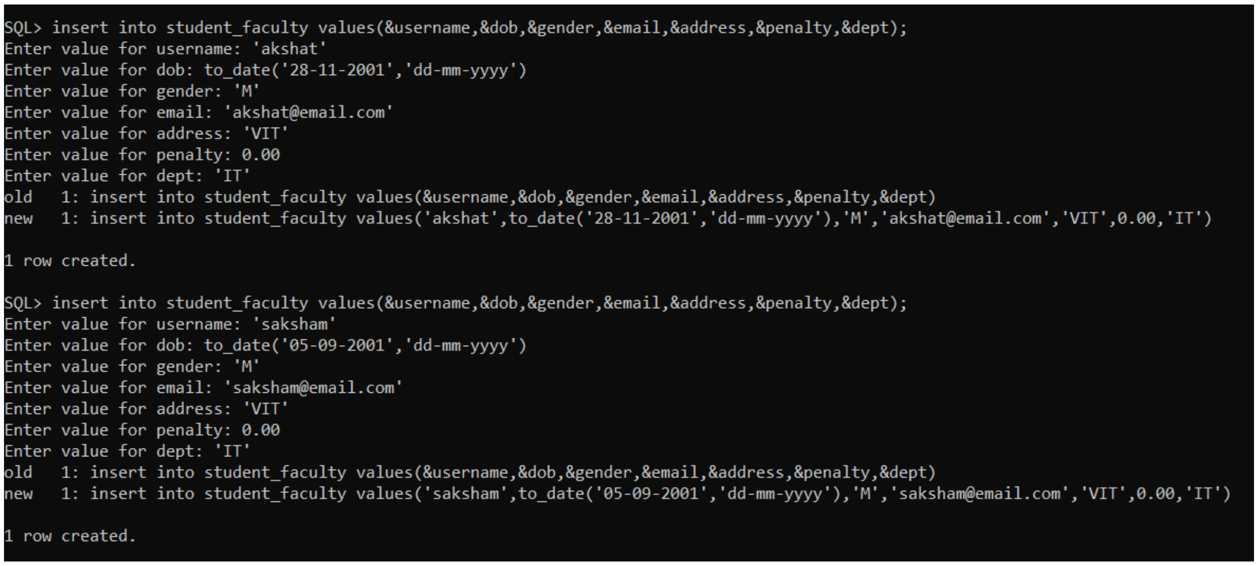
```
SQL> select * from luser;

USERNAME        PASSWORD             NAME              ISSTAFF
--------------- -------------------- --------------- ----------
akshat          akshat123            Akshat Mishra            0
saksham         saksham123           Saksham Arora            0
swetank         swetank123           Swetank                  1
shashank        shashank123          Shashank                 1
```

STAFF

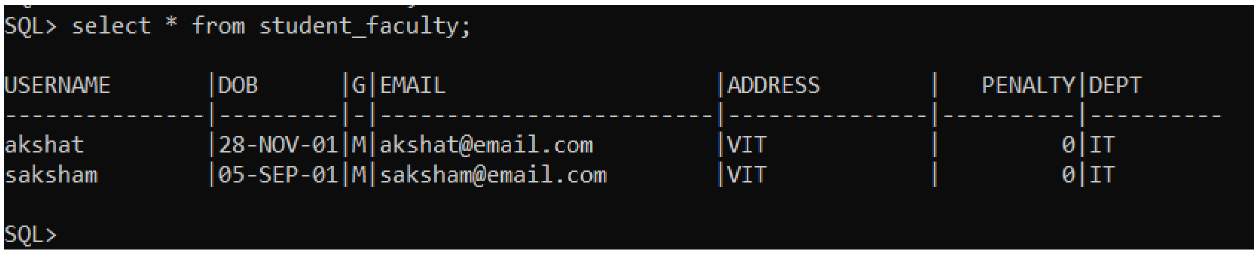


```
SQL> insert into staff values(&username,&staff_id,&job_type,&salary);
Enter value for username: 'shashank'
Enter value for staff_id: 1001
Enter value for job_type: 'Librarian'
Enter value for salary: 1500
old   1: insert into staff values(&username,&staff_id,&job_type,&salary)
new   1: insert into staff values('shashank',1001,'Librarian',1500)

1 row created.

SQL> insert into staff values(&username,&staff_id,&job_type,&salary);
Enter value for username: 'swetank'
Enter value for staff_id: 1002
Enter value for job_type: 'Director'
Enter value for salary: 500000
old   1: insert into staff values(&username,&staff_id,&job_type,&salary)
new   1: insert into staff values('swetank',1002,'Director',500000)

1 row created.

SQL> select * from staff;

USERNAME          STAFF_ID JOB_TYPE                 SALARY
--------------- ---------- -------------------- ----------
shashank              1001 Librarian                  1500
swetank               1002 Director                 500000
```

STUDENT_FACULTY

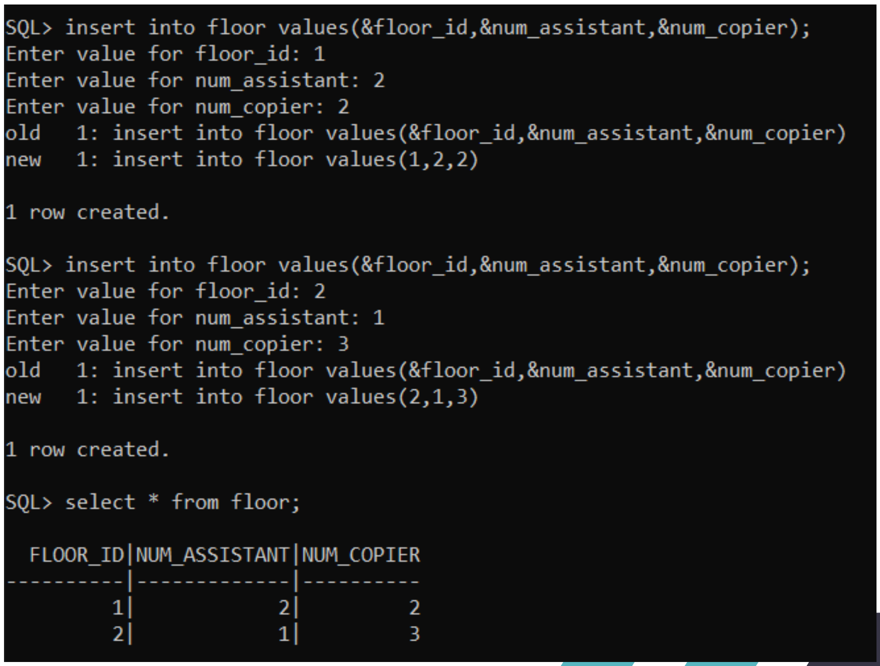


```
SQL> insert into student_faculty values(&username,&dob,&gender,&email,&address,&penalty,&dept);
Enter value for username: 'akshat'
Enter value for dob: to_date('28-11-2001','dd-mm-yyyy')
Enter value for gender: 'M'
Enter value for email: 'akshat@email.com'
Enter value for address: 'VIT'
Enter value for penalty: 0.00
Enter value for dept: 'IT'
old   1: insert into student_faculty values(&username,&dob,&gender,&email,&address,&penalty,&dept)
new   1: insert into student_faculty values('akshat',to_date('28-11-2001','dd-mm-yyyy'),'M','akshat@email.com','VIT',0.00,'IT')

1 row created.

SQL> insert into student_faculty values(&username,&dob,&gender,&email,&address,&penalty,&dept);
Enter value for username: 'saksham'
Enter value for dob: to_date('05-09-2001','dd-mm-yyyy')
Enter value for gender: 'M'
Enter value for email: 'saksham@email.com'
Enter value for address: 'VIT'
Enter value for penalty: 0.00
Enter value for dept: 'IT'
old   1: insert into student_faculty values(&username,&dob,&gender,&email,&address,&penalty,&dept)
new   1: insert into student_faculty values('saksham',to_date('05-09-2001','dd-mm-yyyy'),'M','saksham@email.com','VIT',0.00,'IT')

1 row created.
```



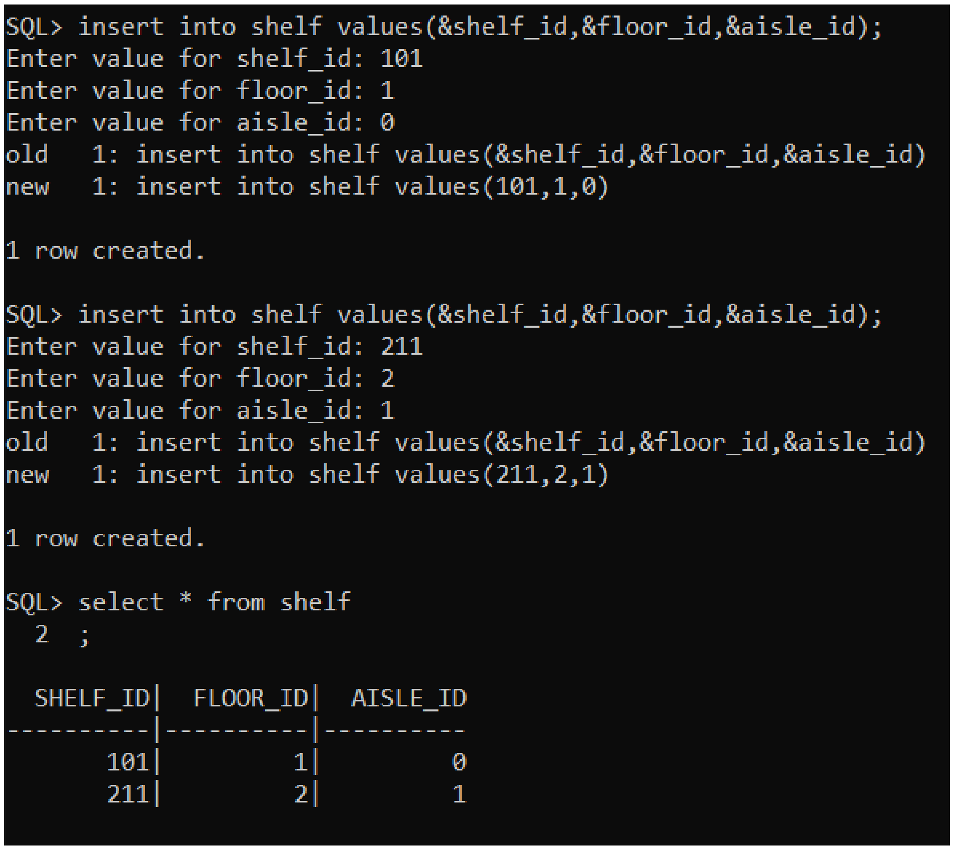
```
SQL> select * from student_faculty;

USERNAME       |DOB      |G|EMAIL                    |ADDRESS        |   PENALTY|DEPT
---------------|---------|-|-------------------------|---------------|----------|----------
akshat         |28-NOV-01|M|akshat@email.com         |VIT            |         0|IT
saksham        |05-SEP-01|M|saksham@email.com        |VIT            |         0|IT

SQL>
```

FLOOR



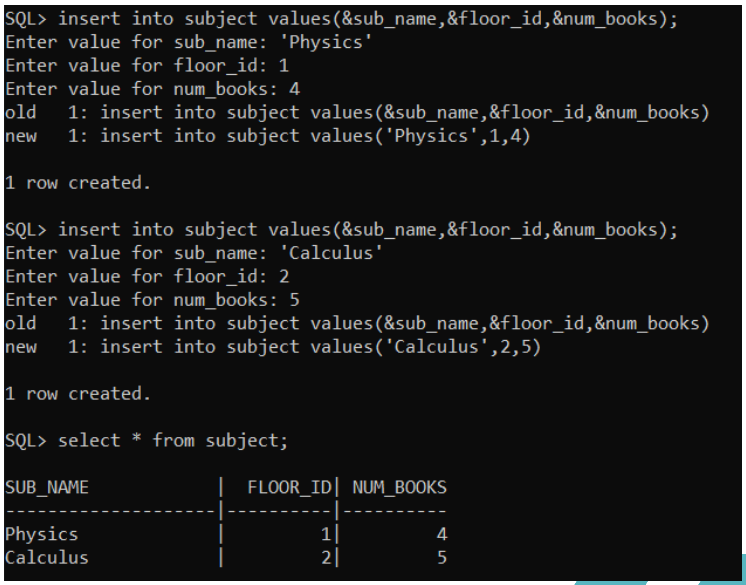
```
SQL> insert into floor values(&floor_id,&num_assistant,&num_copier);
Enter value for floor_id: 1
Enter value for num_assistant: 2
Enter value for num_copier: 2
old   1: insert into floor values(&floor_id,&num_assistant,&num_copier)
new   1: insert into floor values(1,2,2)

1 row created.

SQL> insert into floor values(&floor_id,&num_assistant,&num_copier);
Enter value for floor_id: 2
Enter value for num_assistant: 1
Enter value for num_copier: 3
old   1: insert into floor values(&floor_id,&num_assistant,&num_copier)
new   1: insert into floor values(2,1,3)

1 row created.

SQL> select * from floor;

  FLOOR_ID|NUM_ASSISTANT|NUM_COPIER
----------|-------------|----------
         1|            2|         2
         2|            1|         3
```

SHELF



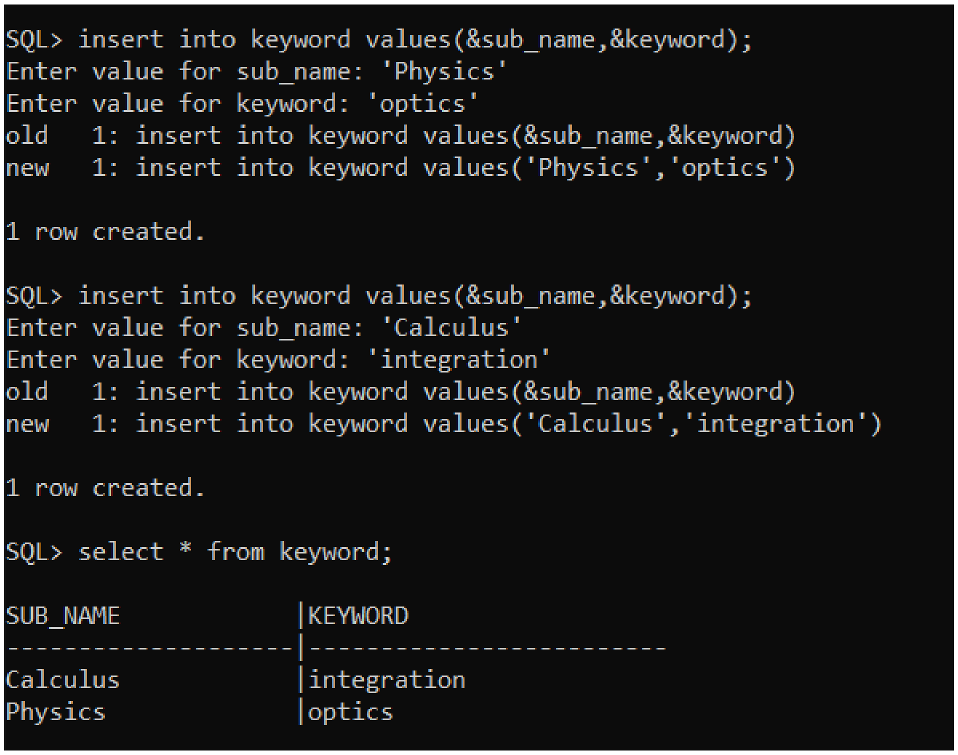
```
SQL> insert into shelf values(&shelf_id,&floor_id,&aisle_id);
Enter value for shelf_id: 101
Enter value for floor_id: 1
Enter value for aisle_id: 0
old   1: insert into shelf values(&shelf_id,&floor_id,&aisle_id)
new   1: insert into shelf values(101,1,0)

1 row created.

SQL> insert into shelf values(&shelf_id,&floor_id,&aisle_id);
Enter value for shelf_id: 211
Enter value for floor_id: 2
Enter value for aisle_id: 1
old   1: insert into shelf values(&shelf_id,&floor_id,&aisle_id)
new   1: insert into shelf values(211,2,1)

1 row created.

SQL> select * from shelf
  2  ;

  SHELF_ID|  FLOOR_ID|  AISLE_ID
----------|----------|----------
       101|         1|         0
       211|         2|         1
```

SUBJECT



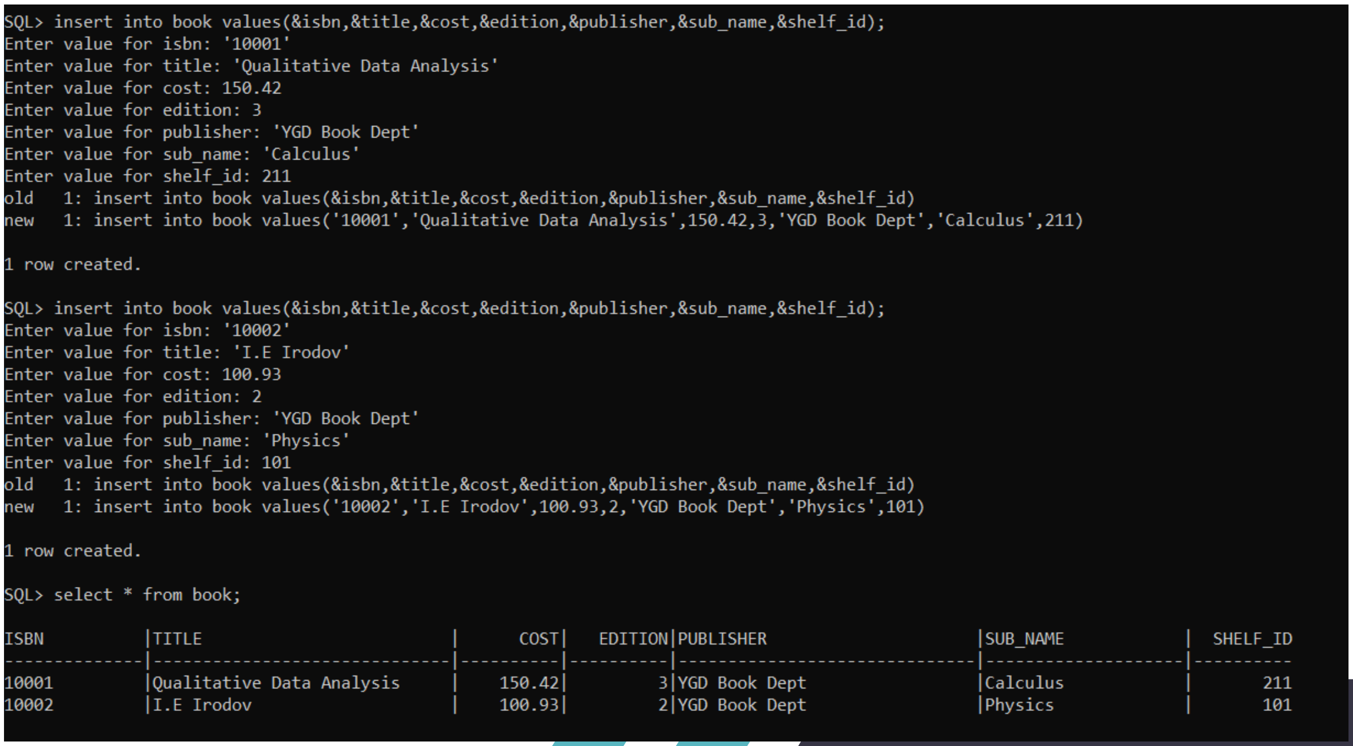
```
SQL> insert into subject values(&sub_name,&floor_id,&num_books);
Enter value for sub_name: 'Physics'
Enter value for floor_id: 1
Enter value for num_books: 4
old   1: insert into subject values(&sub_name,&floor_id,&num_books)
new   1: insert into subject values('Physics',1,4)

1 row created.

SQL> insert into subject values(&sub_name,&floor_id,&num_books);
Enter value for sub_name: 'Calculus'
Enter value for floor_id: 2
Enter value for num_books: 5
old   1: insert into subject values(&sub_name,&floor_id,&num_books)
new   1: insert into subject values('Calculus',2,5)

1 row created.

SQL> select * from subject;

SUB_NAME            |  FLOOR_ID| NUM_BOOKS
--------------------|----------|----------
Physics             |         1|         4
Calculus            |         2|         5
```

KEYWORD



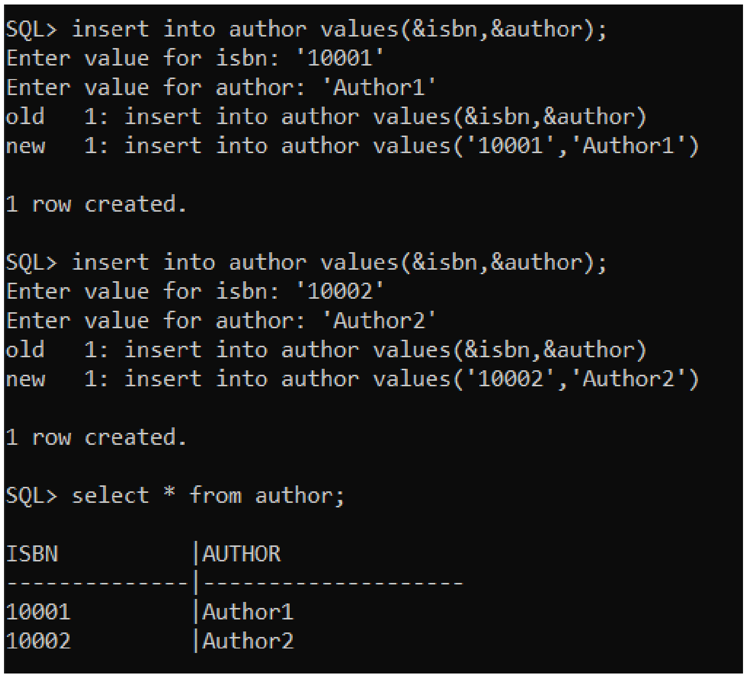
```
SQL> insert into keyword values(&sub_name,&keyword);
Enter value for sub_name: 'Physics'
Enter value for keyword: 'optics'
old   1: insert into keyword values(&sub_name,&keyword)
new   1: insert into keyword values('Physics','optics')

1 row created.

SQL> insert into keyword values(&sub_name,&keyword);
Enter value for sub_name: 'Calculus'
Enter value for keyword: 'integration'
old   1: insert into keyword values(&sub_name,&keyword)
new   1: insert into keyword values('Calculus','integration')

1 row created.

SQL> select * from keyword;

SUB_NAME            |KEYWORD
--------------------|-------------------------
Calculus            |integration
Physics             |optics
```

BOOK



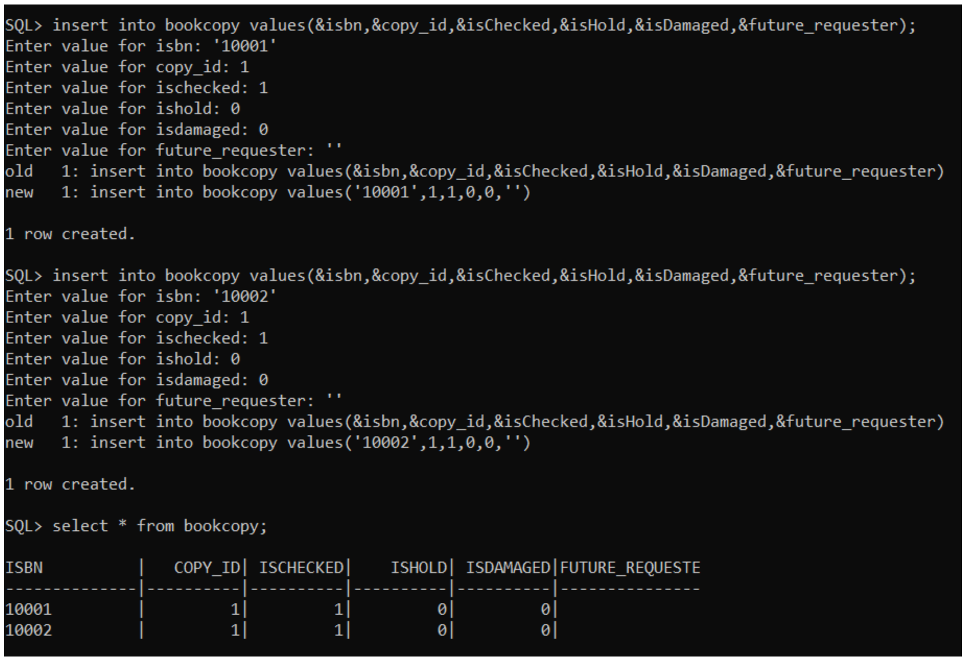
```
SQL> insert into book values(&isbn,&title,&cost,&edition,&publisher,&sub_name,&shelf_id);
Enter value for isbn: '10001'
Enter value for title: 'Qualitative Data Analysis'
Enter value for cost: 150.42
Enter value for edition: 3
Enter value for publisher: 'YGD Book Dept'
Enter value for sub_name: 'Calculus'
Enter value for shelf_id: 211
old   1: insert into book values(&isbn,&title,&cost,&edition,&publisher,&sub_name,&shelf_id)
new   1: insert into book values('10001','Qualitative Data Analysis',150.42,3,'YGD Book Dept','Calculus',211)

1 row created.

SQL> insert into book values(&isbn,&title,&cost,&edition,&publisher,&sub_name,&shelf_id);
Enter value for isbn: '10002'
Enter value for title: 'I.E Irodov'
Enter value for cost: 100.93
Enter value for edition: 2
Enter value for publisher: 'YGD Book Dept'
Enter value for sub_name: 'Physics'
Enter value for shelf_id: 101
old   1: insert into book values(&isbn,&title,&cost,&edition,&publisher,&sub_name,&shelf_id)
new   1: insert into book values('10002','I.E Irodov',100.93,2,'YGD Book Dept','Physics',101)

1 row created.

SQL> select * from book;

ISBN          |TITLE                         |      COST|   EDITION|PUBLISHER                     |SUB_NAME            |  SHELF_ID
--------------|------------------------------|----------|----------|------------------------------|--------------------|----------
10001         |Qualitative Data Analysis     |    150.42|         3|YGD Book Dept                 |Calculus            |       211
10002         |I.E Irodov                    |    100.93|         2|YGD Book Dept                 |Physics             |       101
```

AUTHOR



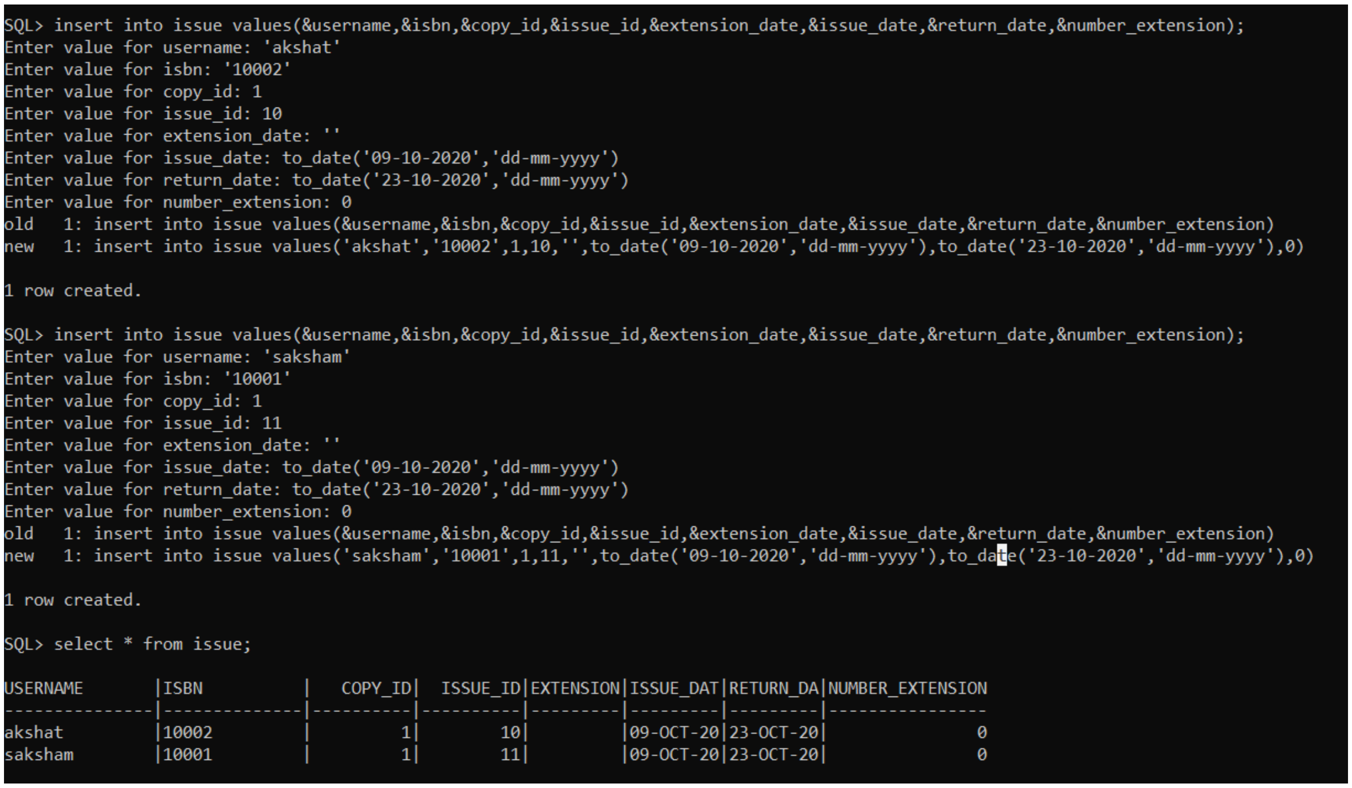
```
SQL> insert into author values(&isbn,&author);
Enter value for isbn: '10001'
Enter value for author: 'Author1'
old   1: insert into author values(&isbn,&author)
new   1: insert into author values('10001','Author1')

1 row created.

SQL> insert into author values(&isbn,&author);
Enter value for isbn: '10002'
Enter value for author: 'Author2'
old   1: insert into author values(&isbn,&author)
new   1: insert into author values('10002','Author2')

1 row created.

SQL> select * from author;

ISBN          |AUTHOR
--------------|--------------------
10001         |Author1
10002         |Author2
```

BOOKCOPY



```
SQL> insert into bookcopy values(&isbn,&copy_id,&isChecked,&isHold,&isDamaged,&future_requester);
Enter value for isbn: '10001'
Enter value for copy_id: 1
Enter value for ischecked: 1
Enter value for ishold: 0
Enter value for isdamaged: 0
Enter value for future_requester: ''
old   1: insert into bookcopy values(&isbn,&copy_id,&isChecked,&isHold,&isDamaged,&future_requester)
new   1: insert into bookcopy values('10001',1,1,0,0,'')

1 row created.

SQL> insert into bookcopy values(&isbn,&copy_id,&isChecked,&isHold,&isDamaged,&future_requester);
Enter value for isbn: '10002'
Enter value for copy_id: 1
Enter value for ischecked: 1
Enter value for ishold: 0
Enter value for isdamaged: 0
Enter value for future_requester: ''
old   1: insert into bookcopy values(&isbn,&copy_id,&isChecked,&isHold,&isDamaged,&future_requester)
new   1: insert into bookcopy values('10002',1,1,0,0,'')

1 row created.

SQL> select * from bookcopy;

ISBN          |   COPY_ID| ISCHECKED|    ISHOLD| ISDAMAGED|FUTURE_REQUESTE
--------------|----------|----------|----------|----------|---------------
10001         |         1|         1|         0|         0|
10002         |         1|         1|         0|         0|
```

ISSUE



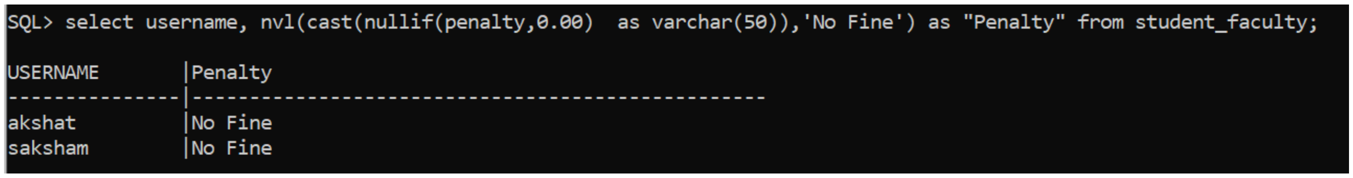
```
SQL> insert into issue values(&username,&isbn,&copy_id,&issue_id,&extension_date,&issue_date,&return_date,&number_extension);
Enter value for username: 'akshat'
Enter value for isbn: '10002'
Enter value for copy_id: 1
Enter value for issue_id: 10
Enter value for extension_date: ''
Enter value for issue_date: to_date('09-10-2020','dd-mm-yyyy')
Enter value for return_date: to_date('23-10-2020','dd-mm-yyyy')
Enter value for number_extension: 0
old   1: insert into issue values(&username,&isbn,&copy_id,&issue_id,&extension_date,&issue_date,&return_date,&number_extension)
new   1: insert into issue values('akshat','10002',1,10,'',to_date('09-10-2020','dd-mm-yyyy'),to_date('23-10-2020','dd-mm-yyyy'),0)

1 row created.

SQL> insert into issue values(&username,&isbn,&copy_id,&issue_id,&extension_date,&issue_date,&return_date,&number_extension);
Enter value for username: 'saksham'
Enter value for isbn: '10001'
Enter value for copy_id: 1
Enter value for issue_id: 11
Enter value for extension_date: ''
Enter value for issue_date: to_date('09-10-2020','dd-mm-yyyy')
Enter value for return_date: to_date('23-10-2020','dd-mm-yyyy')
Enter value for number_extension: 0
old   1: insert into issue values(&username,&isbn,&copy_id,&issue_id,&extension_date,&issue_date,&return_date,&number_extension)
new   1: insert into issue values('saksham','10001',1,11,'',to_date('09-10-2020','dd-mm-yyyy'),to_date('23-10-2020','dd-mm-yyyy'),0)

1 row created.

SQL> select * from issue;

USERNAME       |ISBN          |   COPY_ID|  ISSUE_ID|EXTENSION|ISSUE_DAT|RETURN_DA|NUMBER_EXTENSION
---------------|--------------|----------|----------|---------|---------|---------|----------------
akshat         |10002         |         1|        10|         |09-OCT-20|23-OCT-20|               0
saksham        |10001         |         1|        11|         |09-OCT-20|23-OCT-20|               0
```

REVIEW – 3

6. Write down the necessary SQL statements for implementation of functional requirements (refer to 2) through SQL select, delete and update statement. You may have to modify functional requirements to enable you write complex SQL statements. The SQL statements must include one query showing the usage of nvl function and nullif function, one join query involving order by clause, one uncorrelated nested query, one correlated nested query, one query involving one of the set operators, one query involving group by, having and where clause and one query involving (left or right or full) outer join.

i) One query showing the usage of nvl function and nullif function

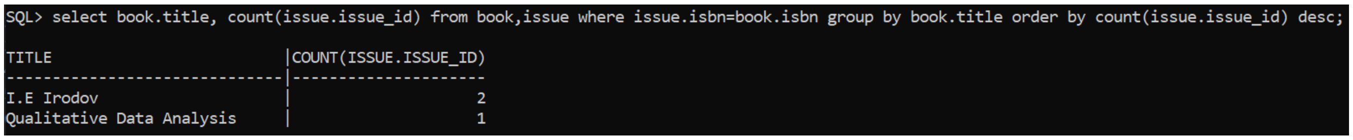
```sql
/* Fine Details */
select username, nvl(cast(nullif(penalty,0.00)  as varchar(50)),'No Fine')
as "Penalty" from student_faculty;
```



```
SQL> select username, nvl(cast(nullif(penalty,0.00)  as varchar(50)),'No Fine') as "Penalty" from student_faculty;

USERNAME       |Penalty
---------------|--------------------------------------------------
akshat         |No Fine
saksham        |No Fine
```

ii) one join query involving order by clause

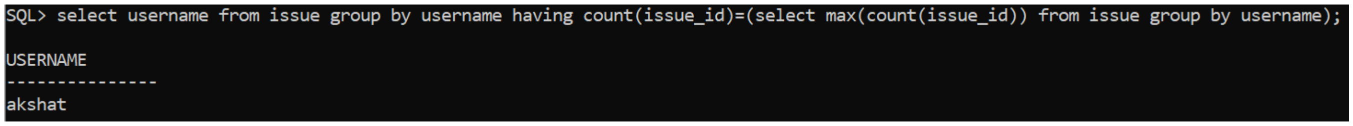
```sql
/* popular book */

select book.title, count(issue.issue_id) from book,issue where issue.isbn=book.isbn gro
up by book.title order by count(issue.issue_id) desc;
```



```
SQL> select book.title, count(issue.issue_id) from book,issue where issue.isbn=book.isbn group by book.title order by count(issue.issue_id) desc;

TITLE                         |COUNT(ISSUE.ISSUE_ID)
------------------------------|---------------------
I.E Irodov                    |                    2
Qualitative Data Analysis     |                    1
```

iii) One uncorrelated nested query

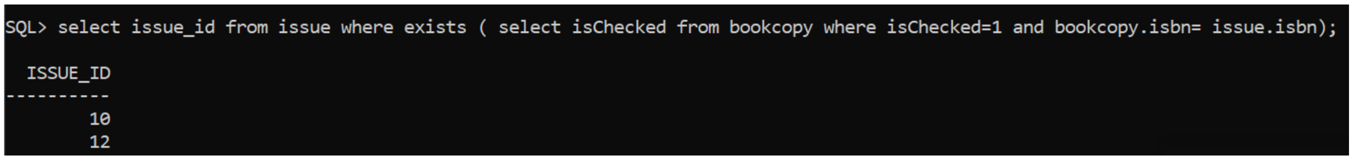
```sql
/* Frequent User */
select username from issue group by username having count(issue_id)=(select max(count(i
ssue_id)) from issue group by username);
```



```
SQL> select username from issue group by username having count(issue_id)=(select max(count(issue_id)) from issue group by username);

USERNAME
---------------
akshat
```

iv) Correlated nested query

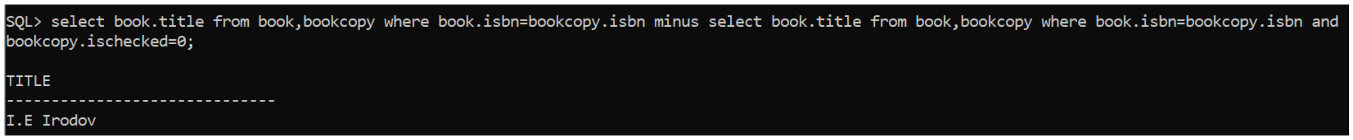
```
/* Books which haven't been returned yet */
select issue_id from issue where exists ( select isChecked from bookcopy where isChecke
d=1 and bookcopy.isbn= issue.isbn);
```



```
SQL> select issue_id from issue where exists ( select isChecked from bookcopy where isChecked=1 and bookcopy.isbn= issue.isbn);

  ISSUE_ID
----------
        10
        12
```

v)   One query involving one of the set operators

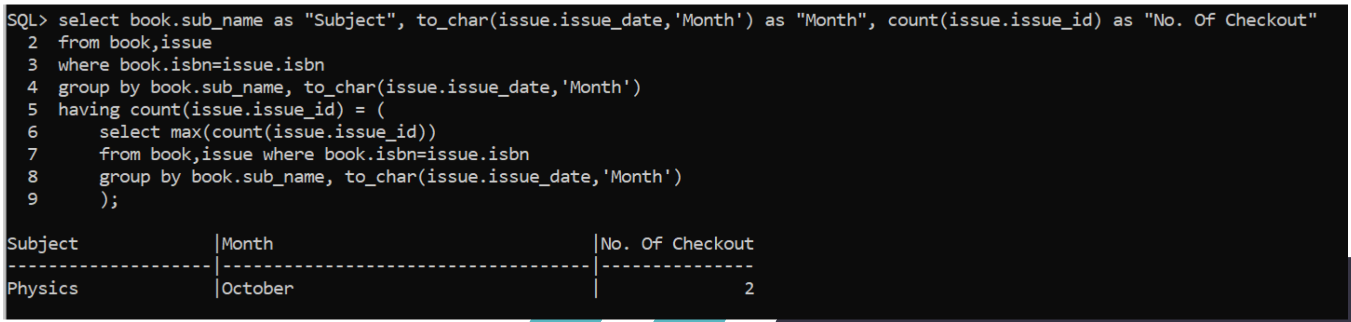
```
/* Checked out Book Copies*/
select book.title from book,bookcopy where book.isbn=bookcopy.isbn minus all select boo
k.title from book,bookcopy where book.isbn=bookcopy.isbn and bookcopy.is_checked=0;
```



```
SQL> select book.title from book,bookcopy where book.isbn=bookcopy.isbn minus select book.title from book,bookcopy where book.isbn=bookcopy.isbn and
bookcopy.ischecked=0;

TITLE
----------------------------
I.E Irodov
```

vi) One query involving group by, having and where clause

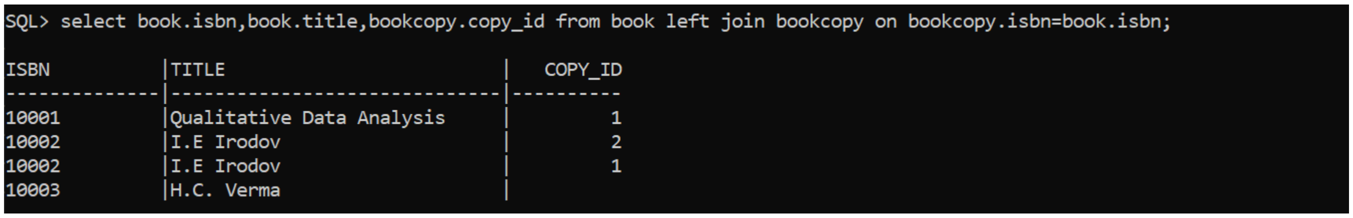
```
select book.sub_name as "Subject", to_char(issue.issue_date,'Month') as "Month", count(
issue.issue_id) as "No. Of Checkout"
from book,issue
where book.isbn=issue.isbn
group by book.sub_name, to_char(issue.issue_date,'Month')
having count(issue.issue_id) = (
    select max(count(issue.issue_id))
    from book,issue where book.isbn=issue.isbn
    group by book.sub_name, to_char(issue.issue_date,'Month')
    );
```



```
SQL> select book.sub_name as "Subject", to_char(issue.issue_date,'Month') as "Month", count(issue.issue_id) as "No. Of Checkout"
     from book,issue
     where book.isbn=issue.isbn
     group by book.sub_name, to_char(issue.issue_date,'Month')
     having count(issue.issue_id) = (
         select max(count(issue.issue_id))
         from book,issue where book.isbn=issue.isbn
         group by book.sub_name, to_char(issue.issue_date,'Month')
         );

Subject             |Month                               |No. Of Checkout
--------------------|------------------------------------|---------------
Physics             |October                             |              2
```

vii) one query involving (left or right or full) outer join.

```
/* Book name with their copies */
select book.isbn,book.title,bookcopy.copy_id from book left join bookcopy on bookcopy.i
sbn=book.isbn;
```

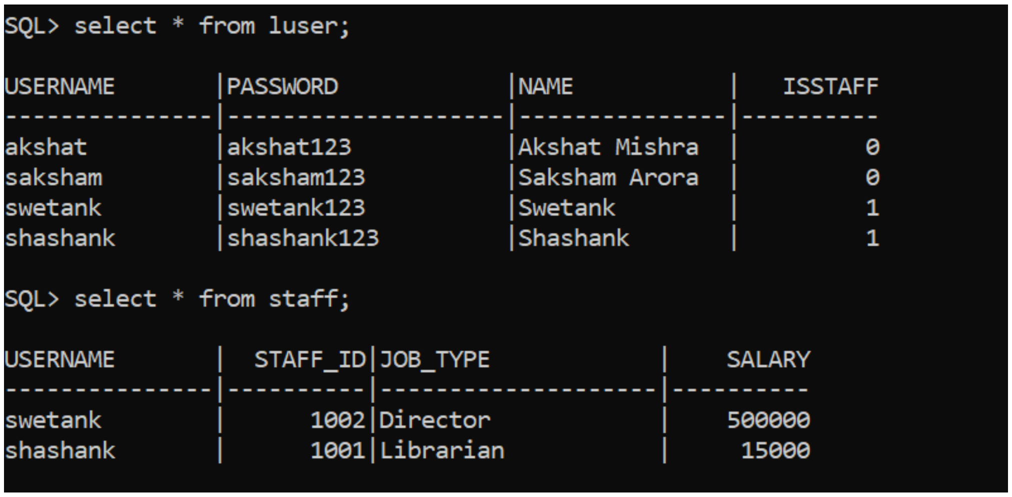


```
SQL> select book.isbn,book.title,bookcopy.copy_id from book left join bookcopy on bookcopy.isbn=book.isbn;

ISBN         |TITLE                        |   COPY_ID
-------------|-----------------------------|----------
10001        |Qualitative Data Analysis    |         1
10002        |I.E Irodov                   |         2
10002        |I.E Irodov                   |         1
10003        |H.C. Verma                   |
```

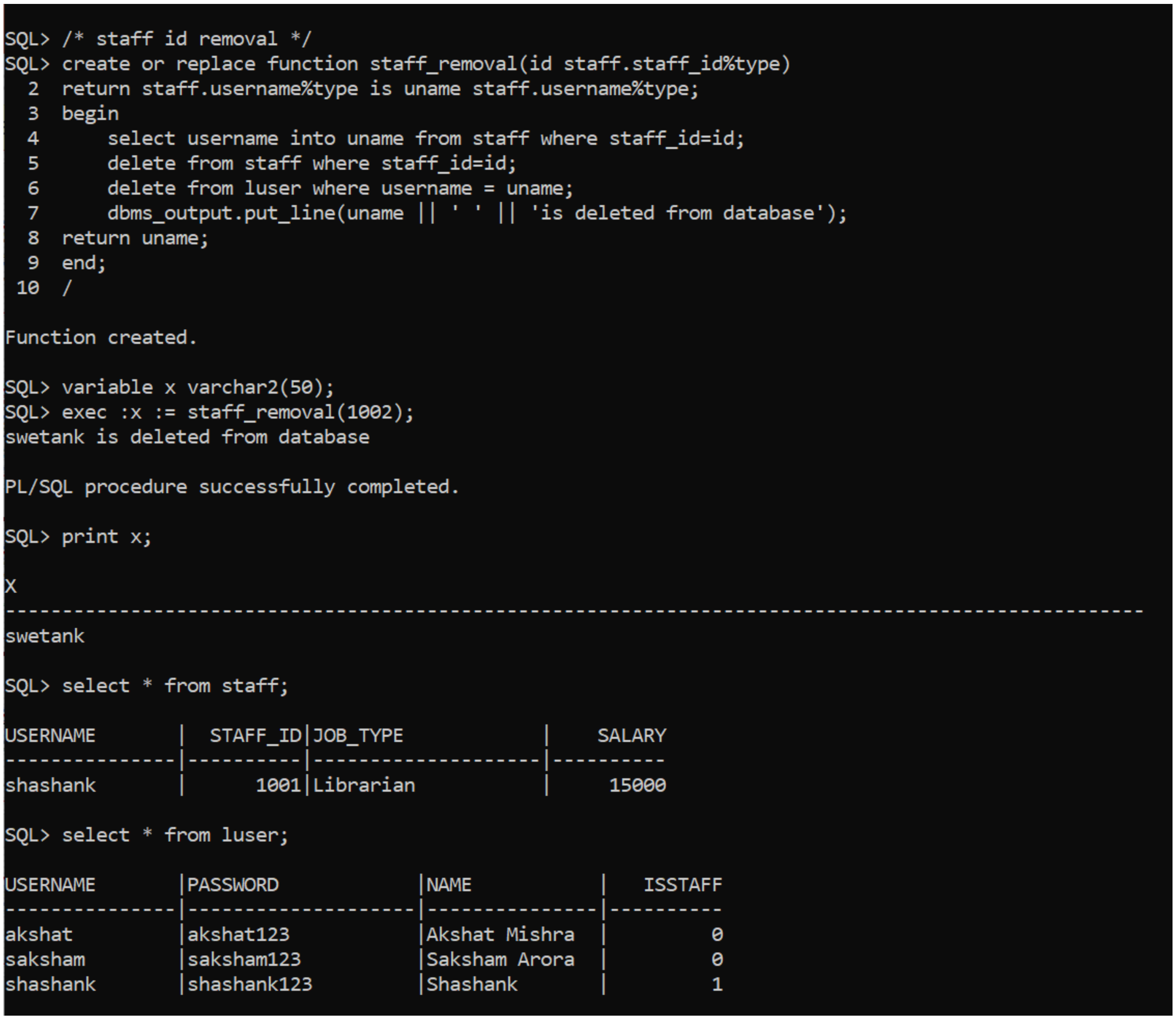
7. Define and implement two PL/SQL function involving cursor and two PL/SQL procedure involving cursor for the database under consideration (i. e. required for the project).

Function 1:

```
/* staff id removal */
create or replace function staff_removal(id staff.staff_id%type)
return staff.username%type is uname staff.username%type;
begin
    select username into uname from staff where staff_id=id;
    delete from staff where staff_id=id;
    delete from luser where username = uname;
    dbms_output.put_line(uname || ' ' || 'is deleted from database');
return uname;
end;
```

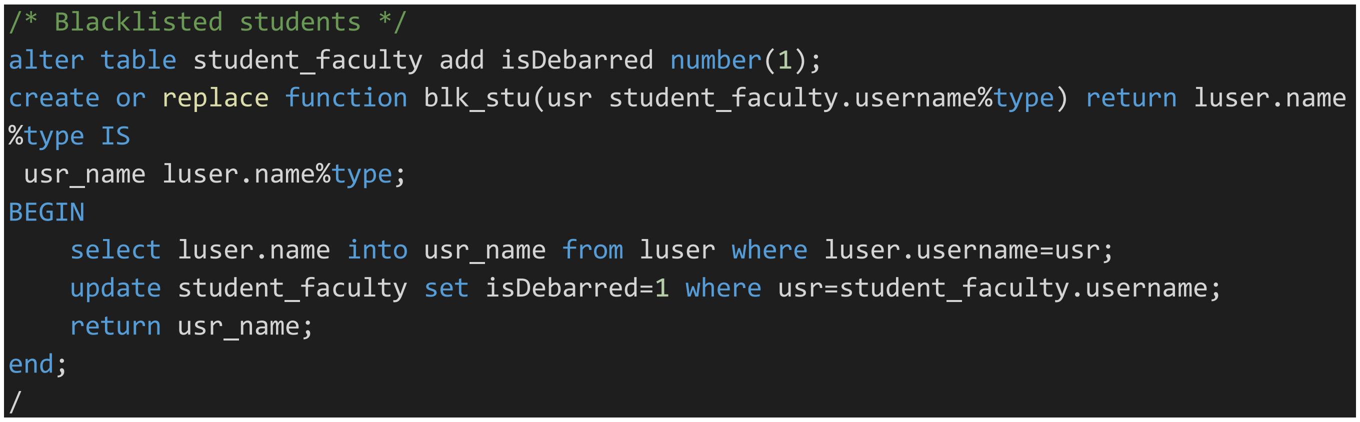


```
SQL> select * from luser;

USERNAME       |PASSWORD            |NAME          |   ISSTAFF
---------------|--------------------|--------------|----------
akshat         |akshat123           |Akshat Mishra |         0
saksham        |saksham123          |Saksham Arora |         0
swetank        |swetank123          |Swetank       |         1
shashank       |shashank123         |Shashank      |         1

SQL> select * from staff;

USERNAME       |   STAFF_ID|JOB_TYPE            |    SALARY
---------------|-----------|--------------------|----------
swetank        |       1002|Director            |    500000
shashank       |       1001|Librarian           |     15000
```

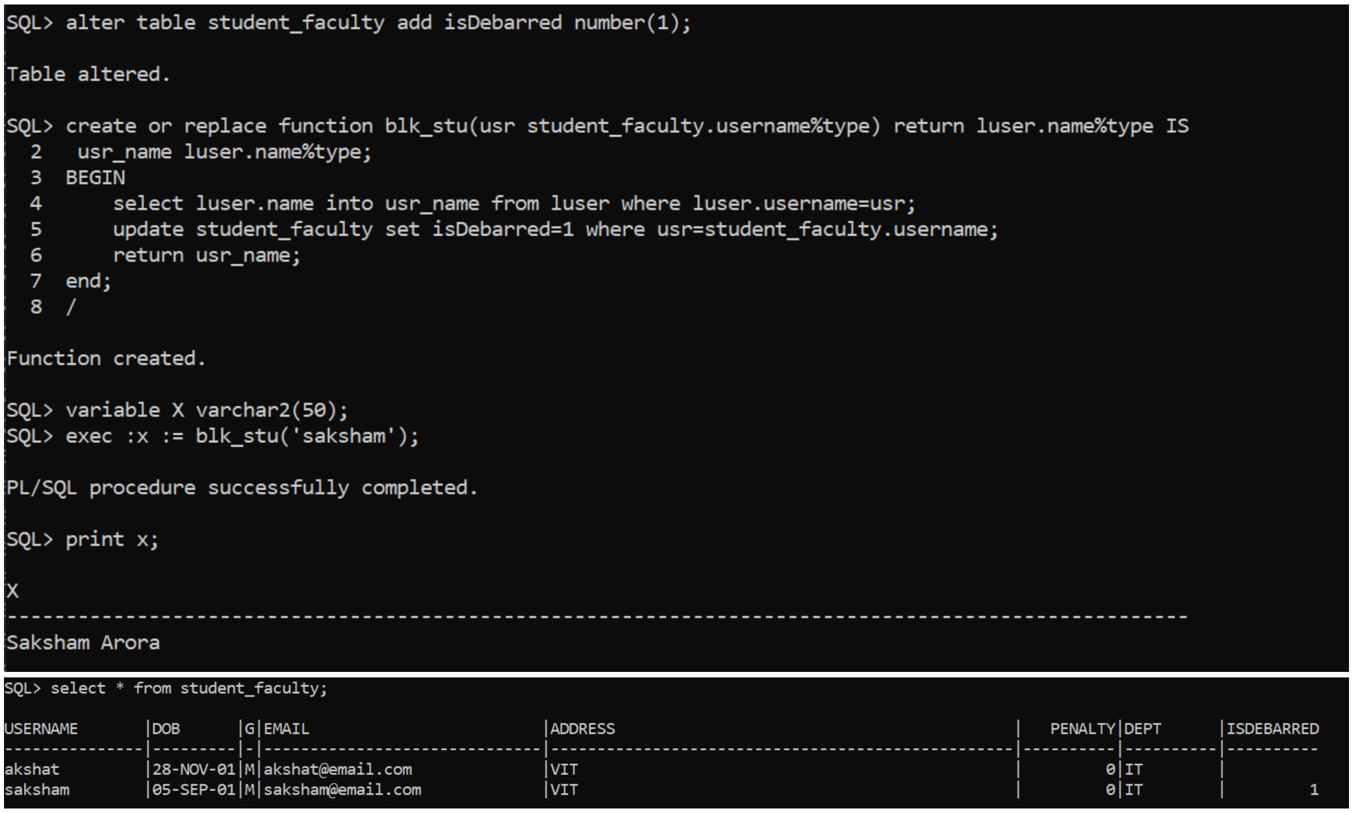


```
SQL> /* staff id removal */
SQL> create or replace function staff_removal(id staff.staff_id%type)
return staff.username%type is uname staff.username%type;
begin
    select username into uname from staff where staff_id=id;
    delete from staff where staff_id=id;
    delete from luser where username = uname;
    dbms_output.put_line(uname || ' ' || 'is deleted from database');
return uname;
end;
/

Function created.

SQL> variable x varchar2(50);
SQL> exec :x := staff_removal(1002);
swetank is deleted from database

PL/SQL procedure successfully completed.

SQL> print x;

X
--------------------------------------------------------------------------------
swetank

SQL> select * from staff;

USERNAME       |  STAFF_ID|JOB_TYPE            |    SALARY
---------------|----------|--------------------|----------
shashank       |      1001|Librarian           |     15000

SQL> select * from luser;

USERNAME       |PASSWORD            |NAME           |   ISSTAFF
---------------|--------------------|---------------|----------
akshat         |akshat123           |Akshat Mishra  |         0
saksham        |saksham123          |Saksham Arora  |         0
shashank       |shashank123         |Shashank       |         1
```

Function 2:



```sql
/* Blacklisted students */
alter table student_faculty add isDebarred number(1);
create or replace function blk_stu(usr student_faculty.username%type) return luser.name
%type IS
 usr_name luser.name%type;
BEGIN
    select luser.name into usr_name from luser where luser.username=usr;
    update student_faculty set isDebarred=1 where usr=student_faculty.username;
    return usr_name;
end;
/
```



```
SQL> alter table student_faculty add isDebarred number(1);

Table altered.

SQL> create or replace function blk_stu(usr student_faculty.username%type) return luser.name%type IS
  2   usr_name luser.name%type;
  3  BEGIN
  4      select luser.name into usr_name from luser where luser.username=usr;
  5      update student_faculty set isDebarred=1 where usr=student_faculty.username;
  6      return usr_name;
  7  end;
  8  /

Function created.

SQL> variable X varchar2(50);
SQL> exec :x := blk_stu('saksham');

PL/SQL procedure successfully completed.

SQL> print x;

X
----------------------------------------------------------------------------------------------------
Saksham Arora
```

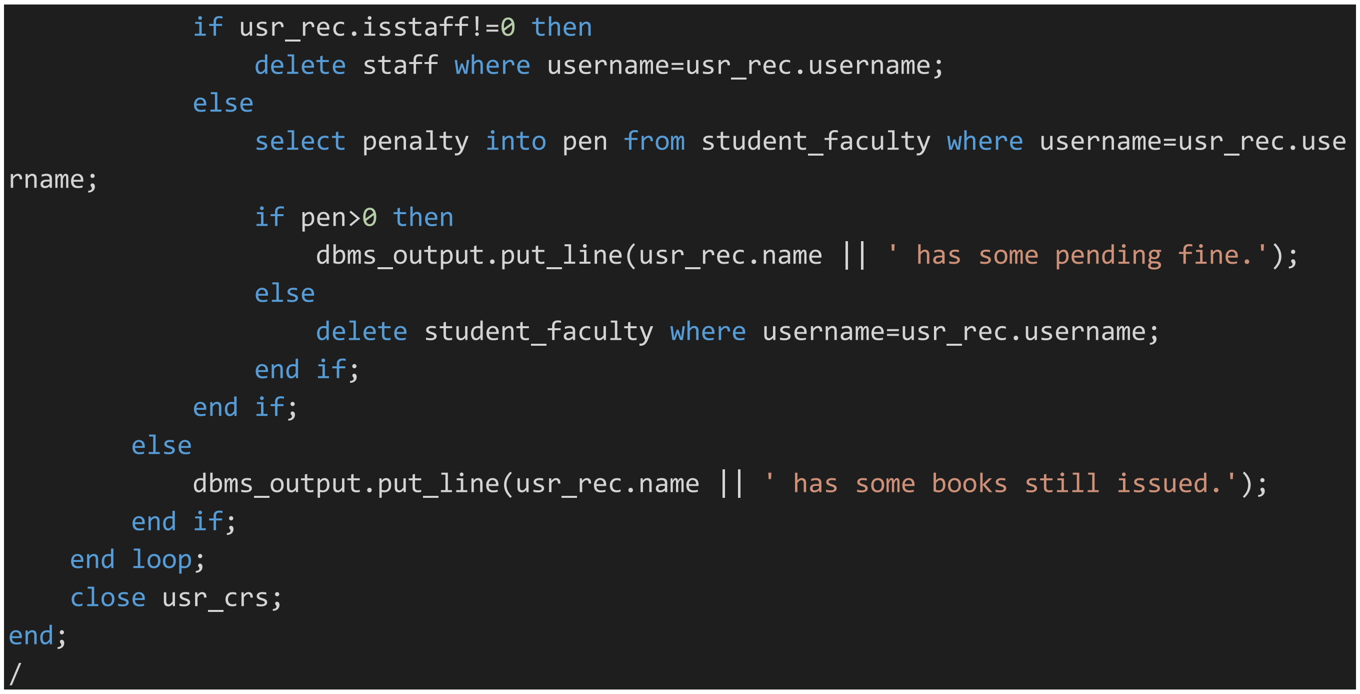
```
SQL> select * from student_faculty;

USERNAME      |DOB      |G|EMAIL                        |ADDRESS                                           |   PENALTY|DEPT      |ISDEBARRED
--------------|---------|-|-----------------------------|--------------------------------------------------|----------|----------|----------
akshat        |28-NOV-01|M|akshat@email.com             |VIT                                               |         0|IT        |
saksham       |05-SEP-01|M|saksham@email.com            |VIT                                               |         0|IT        |         1
```

Procedure 1:

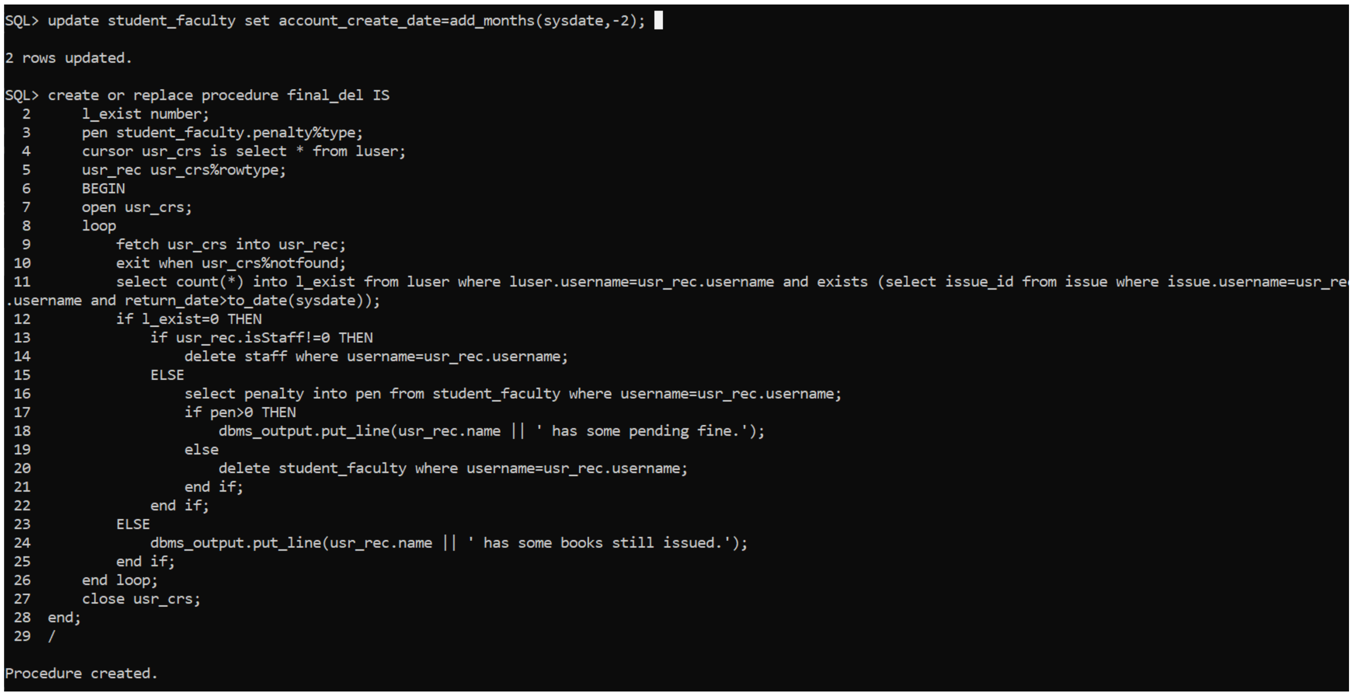
```
/* Extension */
create or replace procedure extend(i_id issue.issue_id%type) IS
    excount issue.number_extension%type;
    staff luser.isStaff%type;
    BEGIN
        select number_extension into excount from issue where issue_id=i_id;
        select isStaff into staff from luser, issue where luser.username=issue.username
 and i_id=issue.id;
        if staff=1 and excount<5 THEN
            update issue set return_date=to_date(return_date)+14,extension_date=to_date
(return_date), number_extension=number_extension+1  where issue_id=i_id;
            dbms_output.put_line('New Extended return date is: '||issue.return_date);
        elsif staff=0 and excount<3 THEN
            update issue set return_date=to_date(return_date)+7,extension_date=to_date(
return_date), number_extension=number_extension+1  where issue_id=i_id;
            dbms_output.put_line('New Extended return date is: '||issue.return_date);
        ELSE
            dbms_output.put_line('all extension counts have been redeemed for this issu
e');
        end if;
    end;
/
```

Procedure 2:

```
alter table student_faculty add account_create_date date;
update student_faculty set account_create_date=add_months(sysdate,-2);
create or replace procedure final_del is
    l_exist number;
    pen student_faculty.penalty%type;
    cursor usr_crs is select * from luser;
    usr_rec usr_crs%rowtype;
    begin
    open usr_crs;
    loop
        fetch usr_crs into usr_rec;
        exit when usr_crs%notfound;
        select count(*) into l_exist from luser where luser.username=usr_rec.username a
nd exists (select issue_id from issue where issue.username=usr_rec.username and return_
date>to_date(sysdate));
        if l_exist=0 then
```



```
            if usr_rec.isstaff!=0 then
                delete staff where username=usr_rec.username;
            else
                select penalty into pen from student_faculty where username=usr_rec.use
rname;
                if pen>0 then
                    dbms_output.put_line(usr_rec.name || ' has some pending fine.');
                else
                    delete student_faculty where username=usr_rec.username;
                end if;
            end if;
        else
            dbms_output.put_line(usr_rec.name || ' has some books still issued.');
        end if;
    end loop;
    close usr_crs;
end;
/
```

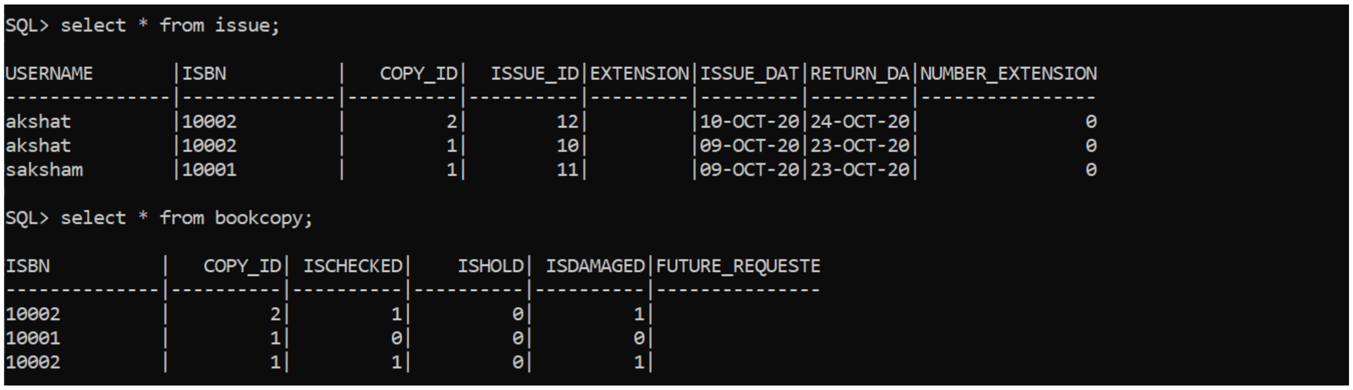


```
SQL> update student_faculty set account_create_date=add_months(sysdate,-2);

2 rows updated.

SQL> create or replace procedure final_del IS
         l_exist number;
         pen student_faculty.penalty%type;
         cursor usr_crs is select * from luser;
         usr_rec usr_crs%rowtype;
         BEGIN
         open usr_crs;
         loop
             fetch usr_crs into usr_rec;
             exit when usr_crs%notfound;
             select count(*) into l_exist from luser where luser.username=usr_rec.username and exists (select issue_id from issue where issue.username=usr_re
.username and return_date>to_date(sysdate));
             if l_exist=0 THEN
                 if usr_rec.isStaff!=0 THEN
                     delete staff where username=usr_rec.username;
                 ELSE
                     select penalty into pen from student_faculty where username=usr_rec.username;
                     if pen>0 THEN
                         dbms_output.put_line(usr_rec.name || ' has some pending fine.');
                     else
                         delete student_faculty where username=usr_rec.username;
                     end if;
                 end if;
             ELSE
                 dbms_output.put_line(usr_rec.name || ' has some books still issued.');
             end if;
         end loop;
         close usr_crs;
     end;
     /

Procedure created.
```

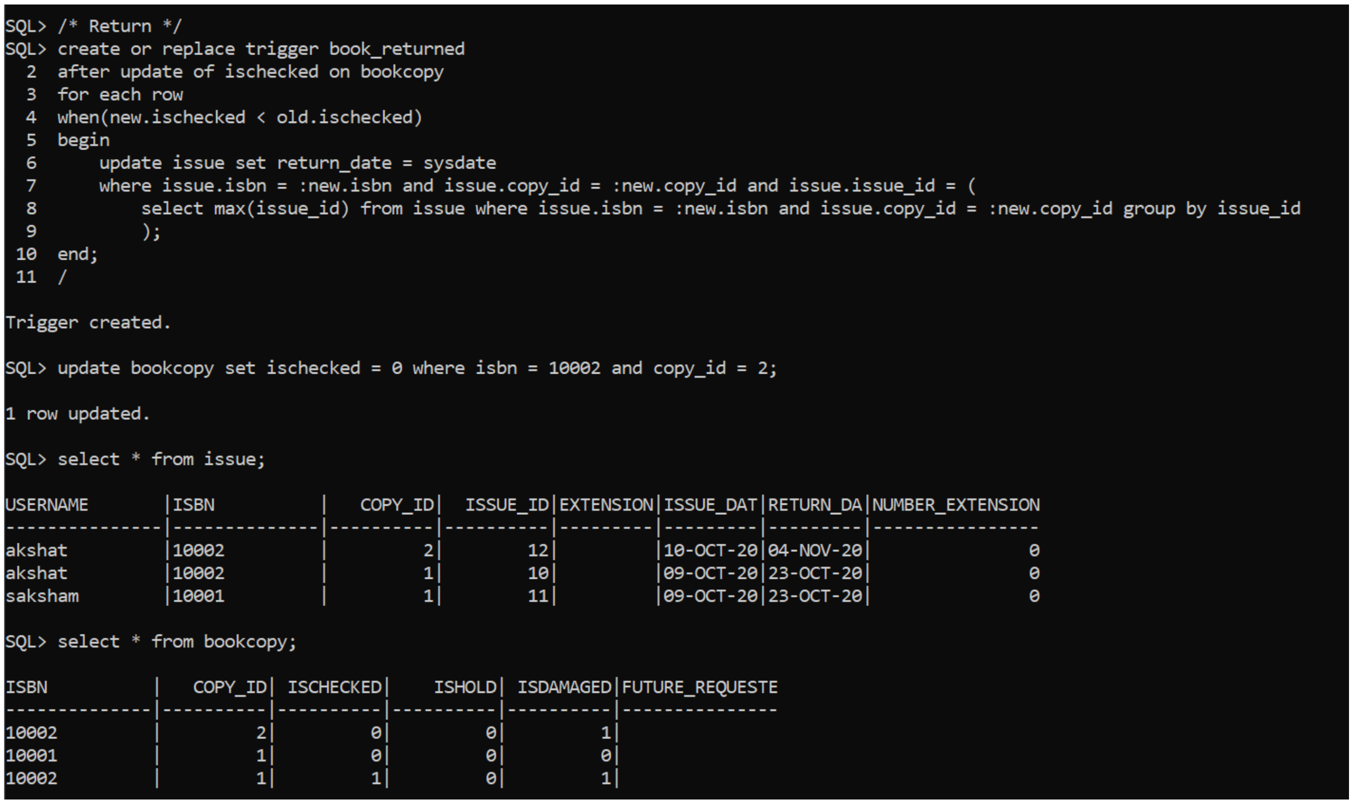
8. Triggers:

Trigger 1:

```
/* Return */
create or replace trigger book_returned
after update of ischecked on bookcopy
for each row
when(new.ischecked < old.ischecked)
begin
    update issue set return_date = sysdate
    where issue.isbn = :new.isbn and issue.copy_id = :new.copy_id and issue.issue_id =
(
        select max(issue_id) from issue where issue.isbn = :new.isbn and issue.copy_id
= :new.copy_id group by issue_id
        );
end;
```



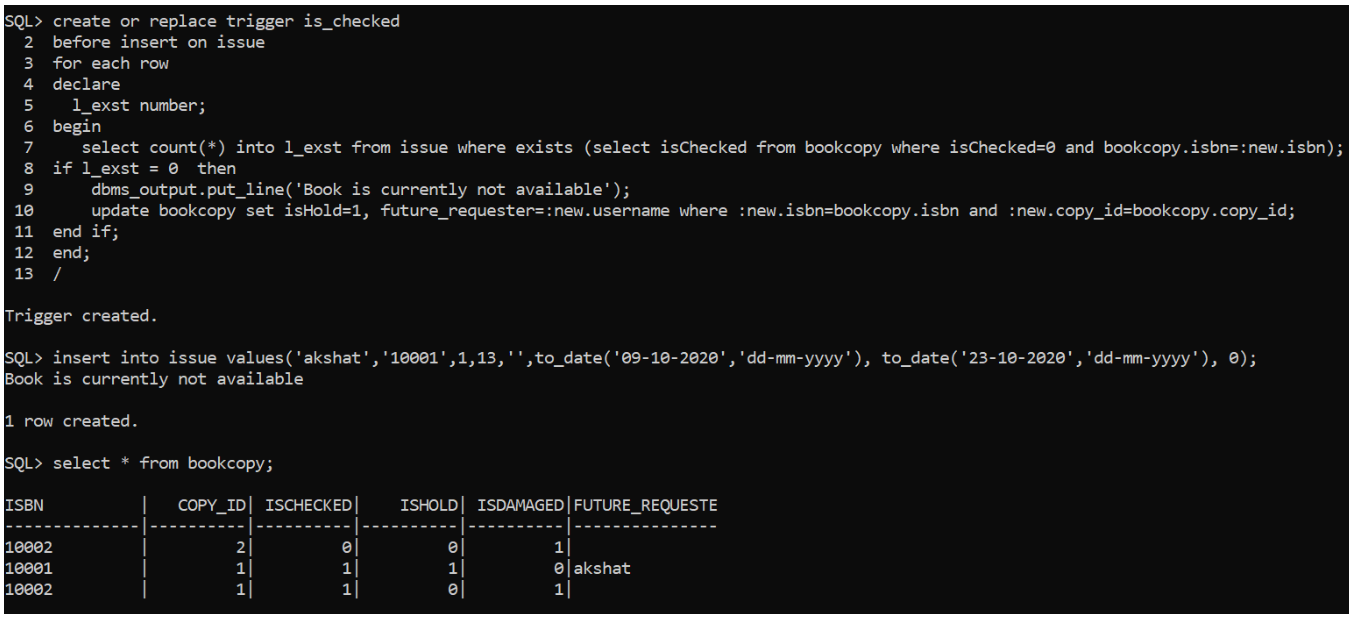
```
SQL> select * from issue;

USERNAME        |ISBN           |   COPY_ID|  ISSUE_ID|EXTENSION|ISSUE_DAT|RETURN_DA|NUMBER_EXTENSION
---------------|--------------|----------|----------|---------|---------|---------|----------------
akshat          |10002          |         2|        12|         |10-OCT-20|24-OCT-20|               0
akshat          |10002          |         1|        10|         |09-OCT-20|23-OCT-20|               0
saksham         |10001          |         1|        11|         |09-OCT-20|23-OCT-20|               0

SQL> select * from bookcopy;

ISBN           |   COPY_ID| ISCHECKED|    ISHOLD| ISDAMAGED|FUTURE_REQUESTE
--------------|----------|----------|----------|----------|---------------
10002          |         2|         1|         0|         1|
10001          |         1|         0|         0|         0|
10002          |         1|         1|         0|         1|
```



```
SQL> /* Return */
SQL> create or replace trigger book_returned
  2  after update of ischecked on bookcopy
  3  for each row
  4  when(new.ischecked < old.ischecked)
  5  begin
  6      update issue set return_date = sysdate
  7      where issue.isbn = :new.isbn and issue.copy_id = :new.copy_id and issue.issue_id = (
  8          select max(issue_id) from issue where issue.isbn = :new.isbn and issue.copy_id = :new.copy_id group by issue_id
  9          );
 10  end;
 11  /

Trigger created.

SQL> update bookcopy set ischecked = 0 where isbn = 10002 and copy_id = 2;

1 row updated.

SQL> select * from issue;

USERNAME       |ISBN          |   COPY_ID|  ISSUE_ID|EXTENSION|ISSUE_DAT|RETURN_DA|NUMBER_EXTENSION
---------------|--------------|----------|----------|---------|---------|---------|----------------
akshat         |10002         |         2|        12|         |10-OCT-20|04-NOV-20|               0
akshat         |10002         |         1|        10|         |09-OCT-20|23-OCT-20|               0
saksham        |10001         |         1|        11|         |09-OCT-20|23-OCT-20|               0

SQL> select * from bookcopy;

ISBN          |   COPY_ID| ISCHECKED|    ISHOLD| ISDAMAGED|FUTURE_REQUESTE
--------------|----------|----------|----------|----------|---------------
10002         |         2|         0|         0|         1|
10001         |         1|         0|         0|         0|
10002         |         1|         1|         0|         1|
```

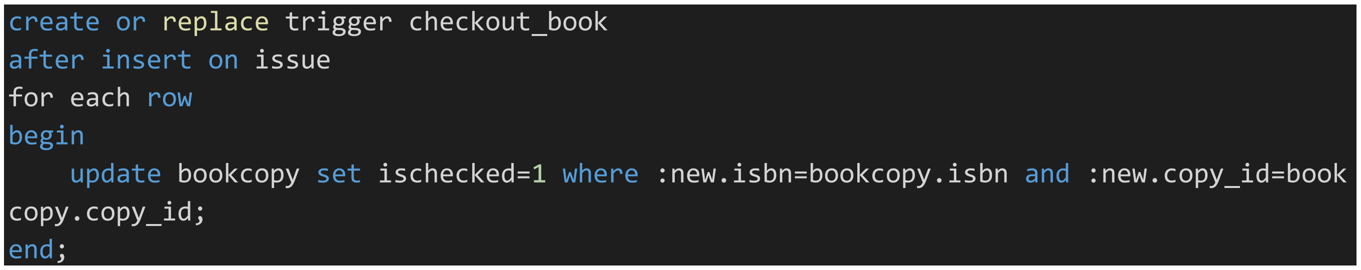
Trigger 2:

```
create or replace trigger is_checked
before insert on issue
for each row
declare
  l_exst number;
begin
   select count(*) into l_exst from issue where exists (select isChecked from bookcopy
where isChecked=0 and bookcopy.isbn=:new.isbn);
if l_exst = 0  then
    dbms_output.put_line('Book is currently not available');
    update bookcopy set isHold=1, future_requester=:new.username where :new.isbn=bookco
py.isbn and :new.copy_id=bookcopy.copy_id;
end if;
end;
```

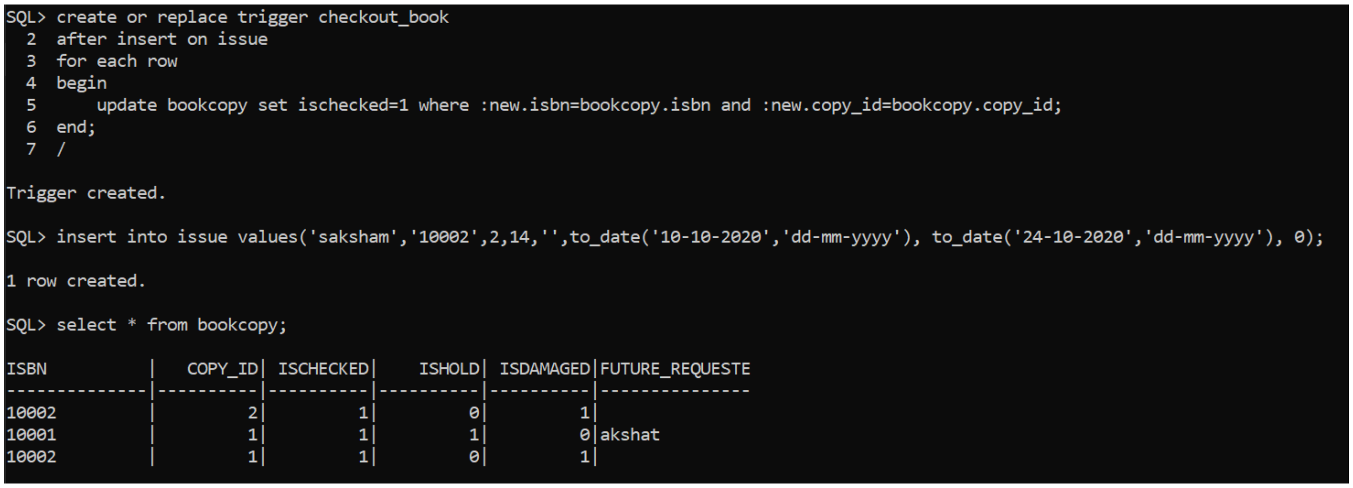


```
SQL> create or replace trigger is_checked
     before insert on issue
     for each row
     declare
       l_exst number;
     begin
        select count(*) into l_exst from issue where exists (select isChecked from bookcopy where isChecked=0 and bookcopy.isbn=:new.isbn);
     if l_exst = 0  then
         dbms_output.put_line('Book is currently not available');
         update bookcopy set isHold=1, future_requester=:new.username where :new.isbn=bookcopy.isbn and :new.copy_id=bookcopy.copy_id;
     end if;
     end;
     /

Trigger created.

SQL> insert into issue values('akshat','10001',1,13,'',to_date('09-10-2020','dd-mm-yyyy'), to_date('23-10-2020','dd-mm-yyyy'), 0);
Book is currently not available

1 row created.

SQL> select * from bookcopy;

ISBN          |   COPY_ID| ISCHECKED|    ISHOLD| ISDAMAGED|FUTURE_REQUESTE
--------------|----------|----------|----------|----------|---------------
10002         |         2|         0|         0|         1|
10001         |         1|         1|         1|         0|akshat
10002         |         1|         1|         0|         1|
```

Trigger 3:



```
create or replace trigger checkout_book
after insert on issue
for each row
begin
    update bookcopy set ischecked=1 where :new.isbn=bookcopy.isbn and :new.copy_id=book
copy.copy_id;
end;
```



```
SQL> create or replace trigger checkout_book
     after insert on issue
     for each row
     begin
         update bookcopy set ischecked=1 where :new.isbn=bookcopy.isbn and :new.copy_id=bookcopy.copy_id;
     end;
     /

Trigger created.

SQL> insert into issue values('saksham','10002',2,14,'',to_date('10-10-2020','dd-mm-yyyy'), to_date('24-10-2020','dd-mm-yyyy'), 0);

1 row created.

SQL> select * from bookcopy;

ISBN          |   COPY_ID| ISCHECKED|    ISHOLD| ISDAMAGED|FUTURE_REQUESTE
--------------|----------|----------|----------|----------|---------------
10002         |         2|         1|         0|         1|
10001         |         1|         1|         1|         0|akshat
10002         |         1|         1|         0|         1|
```