CSCI 411

Project 2

Group V

ER Diagram

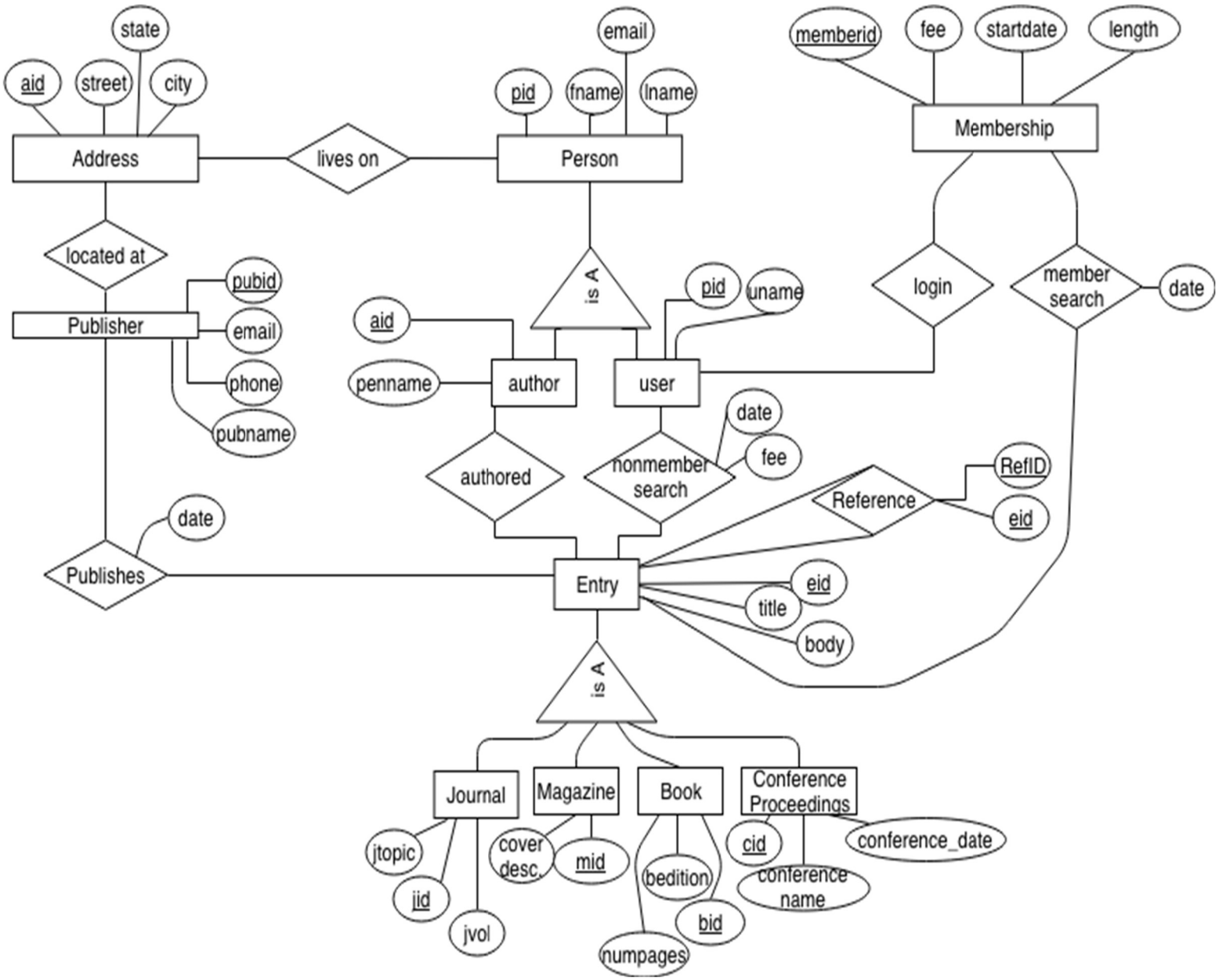

state
aid
street
city
Address
lives on
email
pid
fname
lname
Person
memberid
fee
startdate
length
Membership
located at
Publisher
pubid
email
phone
pubname
is A
pid
uname
aid
penname
author
user
login
member search
date
date
fee
authored
nonmember search
Reference
RefID
eid
date
Publishes
Entry
eid
title
body
is A
Journal
Magazine
Book
Conference Proceedings
jtopic
jid
jvol
cover desc.
mid
bedition
bid
numpages
cid
conference name
conference_date

Table creation SQL and Contents

```
CREATE TABLE Person (
  pid number default NULL,
  fname varchar2(255) default NULL,
  lname varchar2(255) default NULL,
  Email varchar2(255) default NULL,
  PRIMARY KEY (pid)
);

CREATE TABLE Author (
  pid number NOT NULL references person(pid),
  penname varchar2(255),
  PRIMARY KEY (pid)
);

CREATE TABLE Publisher(
  pubid number default NULL,
  pubname varchar2(255) default NULL,
  phone varchar2(255) default NULL,
  email varchar2(255) default NULL,
  PRIMARY KEY (pubid)
);

CREATE TABLE Address (
  aid number NOT NULL,
  street varchar2(255) NOT NULL,
  city varchar2(255) NOT NULL,
  state varchar2(255) NOT NULL,
  PRIMARY KEY(aid)
);

CREATE TABLE Membership(
  memberid varchar2(255),
  length number,
  start_date date,
  fee real NOT NULL,
  PRIMARY KEY (memberid)
);
```

```
CREATE TABLE Users (
  pid number NOT NULL references person(pid),
  uname varchar2(255),
  PRIMARY KEY (pid)
);

CREATE TABLE Entry (
  eid varchar2(255),
  title varchar2(255),
  body varchar2(255),
  PRIMARY KEY (eid)
);

CREATE TABLE book (
        bid varchar2(255),
        bedition integer,
        numpages integer,
        Primary key(bid),
        FOREIGN KEY(bid) REFERENCES Entry(eid) ON DELETE CASCADE
);

CREATE TABLE Journal (
  jid varchar2(255),
  jtopic varchar2(255),
  jvol integer,
  PRIMARY KEY (jid),
  FOREIGN KEY (jid) references Entry(eid) ON DELETE CASCADE
 );

CREATE TABLE Magazine (
  mid varchar2(255),
  cover_description varchar2(255),
  PRIMARY KEY (mid),
  FOREIGN KEY (mid) references Entry(eid) ON DELETE CASCADE
 );



CREATE TABLE Conference_proceedings (
  cid varchar2(255),
```

```
  conference_name varchar2(255),
  conference_date date,
  PRIMARY KEY (cid),
  FOREIGN KEY (cid) references Entry(eid) ON DELETE CASCADE
 );

CREATE TABLE References (
  eid varchar2(255),
  refid varchar2(255),
  PRIMARY KEY (eid, refid),
  FOREIGN KEY (eid) references Entry ON DELETE CASCADE
 );

CREATE TABLE login(
  pid  number,
  memberid  varchar2(255) NOT NULL,
  PRIMARY KEY (memberid, pid),
  FOREIGN KEY (memberid) references Membership,
  FOREIGN KEY (pid) references Users
 );

CREATE TABLE Member_Search (
  memberid varchar2(255),
  eid varchar2(255),
  retrieved date,
  PRIMARY KEY (memberid, eid, retrieved),
  FOREIGN KEY (memberid) references Membership,
  FOREIGN KEY (eid) references Entry ON DELETE CASCADE
 );

CREATE TABLE Non_Member_Search (
  pid number,
  eid varchar2(255),
  search_date date,
  fee real,
  PRIMARY KEY (pid, eid, search_date),
  FOREIGN KEY (pid) references Users,
  FOREIGN KEY (eid) references Entry ON DELETE CASCADE
 );
```

```
CREATE TABLE Authored(
  eid varchar2(255) NOT NULL,
  pid number NOT NULL,
  PRIMARY KEY (eid, pid),
  FOREIGN KEY (eid) references Entry ON DELETE CASCADE,
  FOREIGN KEY (pid) references Author
);

CREATE TABLE Publishes (
  eid varchar2(255) NOT NULL,
  pubid number NOT NULL,
  published date,
  PRIMARY KEY (eid, pubid),
  FOREIGN KEY (eid) references Entry ON DELETE CASCADE,
  FOREIGN KEY (pubid) references Publisher
);

CREATE TABLE Located_at (
  aid number NOT NULL references Address(aid),
  pubid number NOT NULL references Publisher(pubid),
  PRIMARY KEY (aid, pubid),
  FOREIGN KEY (aid) references Address,
  FOREIGN KEY (pubid) references Publisher
);

CREATE TABLE Lives_At (
  aid number NOT NULL references Address(aid),
  pid number NOT NULL references Person(pid),
  PRIMARY KEY (aid, pid),
  FOREIGN KEY (aid) references Address,
  FOREIGN KEY (pid) references Person
);
```

Data Insertion

Insert into Person (pid, fname, lname, email) ->

pid,fname,lname,email
102,Serge,Meriel,smeriel0@nih.gov
103,Pearle,Finley,pfinley1@domainmarket.com
106,Harmonia,Alleburton,halleburton2@vinaora.com
107,Archibaldo,Warnes,awarnes3@buzzfeed.com
108,Margi,Grayston,mgrayston4@ezinearticles.com
117,Trstram,Gear,tgear5@yelp.com
118,Cindy,Dohmer,cdohmer6@hhs.gov
119,Jackquelin,Dysert,jdysert7@reddit.com
120,Aime,Sandcraft,asandcraft8@ucla.edu
121,Reiko,Brettle,rbrettle9@1688.com
122,Morton,Northern,mnortherna@so-net.ne.jp
123,Aubrey,Lacroutz,alacroutzb@sbwire.com
124,Paton,Tuxwell,ptuxwellc@eepurl.com
125,Emmey,Spearman,espearmand@livejournal.com
126,Shir,Hovell,shovelle@angelfire.com
29,Thaxter,Winston,twinstonf@istockphoto.com
131,Andee,Lecount,alecountg@unesco.org
134,Kelley,Lewisham,klewishamh@edublogs.org
135,Saxon,Garrard,sgarrardi@dell.com
141,Isacco,Minocchi,iminocchij@tamu.edu
142,Harmony,Darrell,hdarrellk@xrea.com
145,Sanderson,Sperski,ssperskil@360.cn
146,Korie,Richemont,krichemontm@ustream.tv
147,Marcia,Fergusson,mfergussonn@quantcast.com
150,Fidole,McCullogh,fmccullogho@apache.org
151,Desirae,Stickens,dstickensp@cornell.edu
152,Talya,Mor,tmorq@hubpages.com
153,Mufinella,Goulbourne,mgoulbourner@typepad.com
200,Tim,Jeffreys,tjeffreyss@yahoo.com
201,Mahmud,McAuley,mmcauleyt@usa.gov
202,Caroljean,Menichini,cmenichiniu@typepad.com
203,Missie,Sutherington,msutheringtonv@theatlantic.com

Insert into address(aid, street, city, state) ->

aid,street,city,state
1000,7671 Marcy Point,Oceanside,California
1001,48974 Sunbrook Center,Lincoln,Nebraska
1002,1 Stephen Point,Boulder,Colorado
1003,49967 Lawn Junction,Worcester,Massachusetts
1004,488 Oak Valley Junction,Boise,Idaho
1005,1 Maywood Hill,Atlanta,Georgia
1006,52979 Butterfield Place,Boston,Massachusetts
1007,3 Starling Parkway,Monticello,Minnesota
1008,9119 Village Green Terrace,Colorado Springs,Colorado

1009,4461 Carioca Point,Lansing,Michigan
1010,17 Victoria Crossing,Fresno,California
1011,08 Eliot Circle,Milwaukee,Wisconsin
1012,96 Onsgard Drive,New York City,New York
1013,1797 Grim Place,Schaumburg,Illinois
1014,49775 Calypso Way,Redwood City,California
1015,1 Charing Cross Alley,Santa Ana,California
1016,282 Trailsway Circle,Indianapolis,Indiana
1017,8 Northview Alley,Lawrenceville,Georgia
1018,3323 Kingsford Drive,Seattle,Washington
1019,8156 Ridgeview Crossing,New York City,New York
1020,3810 Sutteridge Parkway,Redwood City,California
1021,58990 Maywood Center,Tampa,Florida
1022,38 Clove Avenue,Minneapolis,Minnesota
1023,7456 Dunning Lane,Savannah,Georgia
1024,2 Glacier Hill Avenue,Washington,District of Columbia
1025,80730 Becker Circle,Charlotte,North Carolina
1026,650 Superior Circle,Phoenix,Arizona
1027,84 Crowley Center,Las Vegas,Nevada
1028,211 Valley Edge Parkway,Minneapolis,Minnesota
1029,335 Spohn Park,Springfield,Illinois
1030,07628 Manley Avenue,Burbank,California
1031,74162 Northport Circle,Oklahoma City,Oklahoma
1032,01213 Blaine Crossing,Philadelphia,Pennsylvania
1033,1 Schiller Trail,Boise,Idaho
1034,6078 Northland Road,Alhambra,California
1035,516 Thierer Lane,Odessa,Texas
1036,12564 Tony Pass,Long Beach,California
1037,68676 Butternut Way,Tucson,Arizona
1038,995 Alpine Crossing,Spartanburg,South Carolina
1039,35 Westend Place,Philadelphia,Pennsylvania
1040,4 Dahle Circle,Portland,Oregon
1041,326 Katie Trail,Des Moines,Iowa
1042,61 Lakewood Alley,Tampa,Florida
1043,11643 Sullivan Center,Schenectady,New York
1044,651 Bobwhite Park,Valdosta,Georgia
1045,85 Helena Street,Minneapolis,Minnesota
1046,94 Miller Avenue,Richmond,Virginia
1047,849 Mccormick Junction,Watertown,Massachusetts
1048,3825 Pawling Trail,San Francisco,California
1049,296 Gale Terrace,Denver,Colorado
1050,9595 Acker Court,Fort Lauderdale,Florida
1051,38304 Crownhardt Hill,Atlanta,Georgia
1052,571 Kenwood Crossing,Amarillo,Texas
1053,479 Vidon Point,North Little Rock,Arkansas
1054,369 Scott Avenue,Charleston,West Virginia
1055,9079 Golf View Hill,Huntington,West Virginia
1056,9607 Kennedy Road,Tucson,Arizona
1057,88169 Bashford Street,Trenton,New Jersey

1058,50 Buhler Lane,Cincinnati,Ohio
1059,689 Jenna Street,Amarillo,Texas

Insert into Entry(eid, title, body) ->

eid,title,body
IX3646,Data Strcutures,The content of the book is Data Strcutures and Algorithms
IB2946,Rohinhood ,The content of the book is life of Robinhood
LH8542,Snowwhite,The content of the book is tales of Snowwhite
NJ6977,Success,The content of the book is about success
CE7059,Get Rich,The content of the book is about getting rich
VM7843,Banking,The content of the book is about banking
YZ6317,American Medical Soceity,Annual journal from American Medical Society (AMS)
AB2755,American Computer Scientist ,Annual Journal of American Computer Scientist (ACS)
OV3250,American Bankers Association ,Annual journal of American Bankers Association (ABA)
ZL9928,Forbes,List of top 10 influencers in the world
UE6668,Time,List of top 10 billionairs in the world
RY2195,People,Lost of most trending clothes in 2018
KQ4175,AI for Healthcare ,Implementation of AI in hospitals
VX8161,Deep Learning for Surgery ,Implmentation of Deep Learning Neural Networks in Surgery
IX8371,Computer Vision for Heart surgery ,Implmentation of Computer Vision

Insert into Publisher (eid, pubid, published) ->

eid,pubid,published
IX3646,37270,20-Dec-2018
IB2946,45671,30-Jan-2008
LH8542,27542,15-Apr-2008
NJ6977,27542,16-Apr-2008
CE7059,27542,17-Apr-2008
VM7843,27542,18-Apr-2008
YZ6317,17588,17-May-2006
AB2755,15470,8-Oct-2005
OV3250,77523,20-Jun-2004
ZL9928,32585,6-Jun-2005
UE6668,79566,8-Mar-2006
RY2195,32527,9-Feb-2002
KQ4175,28954,16-Oct-2001
VX8161,44641,4-May-2009
IX8371,76522,7-May-2009

Insert into conference_proceedings (cid, confernec_name, conference_date) ->

cid,conference_name,conference_date
KQ4175,AI for Healthcare ,31-May-2006
VX8161,Deep Learning for Surgery ,25-Jan-2012

IX8371,Computer Vision for Heart surgery ,18-Apr-2016

Insert into Journal (jid, jvol, jtopic) ->

jid,jvol,jtopic
YZ6317,56,Annual research reports
AB2755,40,Annual research reports
OV3250,75,Annual research reports

Insert into Magazines (mid, cover_description) ->

mid,cover_description
ZL9928,Forbes top 10 influencers
UE6668,Times top 10 billionairs
RY2195,People most trending clothes

Insert into Book

bid,bedition,numpages
IX3646,1,900
IB2946,2,200
LH8542,1,750
NJ6977,3,600
CE7059,5,200
VM7843,8,500

Insert into Author(pid, penname) ->

pid,penname
102,Flo Riquet
103,Gisela Tremayle
106,
107,Kirsten Ridgewell
108,
117,Lily Worsall
118,
119,
120,Vladamir Ruperto

Insert into Authored (eid, pid) ->

eid,pid
IX3646,102
IB2946,103
LH8542,106
NJ6977,107
CE7059,108
VM7843,117
YZ6317,118
AB2755,119
OV3250,120
ZL9928,117
UE6668,118
RY2195,119
KQ4175,120
VX8161,107
IX8371,108

Insert into User (pid, uname) ->

pid,uname
121,kcard0
122,jcisar1
123,smccaskill2
124,gpoulney3
125,anewlove4
126,pconnar5
29,cofogerty6
131,edurrell7
134,mverrico8
135,etheze9
141,nspira
142,bminnisb
145,ksymsonc
146,mhillandd
147,sdrapere
150,tannesleyf
151,bchalcotg
152,dmeuseh
153,wsherringtoni
200,ggarlingej
201,blugsdink
202,lgopsalll
203,gbaudassim

Insert into Reterive (memebrid, eid, retrived) ->

memberid,eid,retrived
5bfb0970fc13ae4b54000000,IX3646,10-Jan-2019
5bfb0970fc13ae4b54000001,IB2946,11-Jan-2019
5bfb0970fc13ae4b54000002,LH8542,12-Jan-2019
5bfb0970fc13ae4b54000003,LH8542,13-Jan-2019
5bfb0970fc13ae4b54000004,LH8542,14-Jan-2019
5bfb0970fc13ae4b54000005,LH8542,15-Jan-2019
5bfb0970fc13ae4b54000006,LH8542,16-Jan-2019
5bfb0970fc13ae4b54000007,UE6668,17-Jan-2019
5bfb0970fc13ae4b54000008,RY2195,18-Jan-2019
5bfb0970fc13ae4b54000009,KQ4175,19-Jan-2019

Insert into References (eid, refid) ->

eid,refid
IX3646,IB2946
IB2946,LH8542
LH8542,OV3250
NJ6977,LH8542
CE7059,LH8543
VM7843,LH8544
YZ6317,OV3250
AB2755,OV3250
OV3250,IX3646
ZL9928,IB2946
UE6668,IB2946
RY2195,IB2946
KQ4175,YZ6317
VX8161,YZ6317
IX8371,YZ6317

Insert into Publishes (eid, pubid, published) ->

eid,pubid,published
IX3646,37270,20-Dec-2018
IB2946,45671,30-Jan-2008
LH8542,27542,15-Apr-2008
NJ6977,27542,16-Apr-2008
CE7059,27542,17-Apr-2008
VM7843,27542,18-Apr-2008
YZ6317,17588,17-May-2006
AB2755,15470,8-Oct-2005
OV3250,77523,20-Jun-2004
ZL9928,32585,6-Jun-2005
UE6668,79566,8-Mar-2006
RY2195,32527,9-Feb-2002
KQ4175,28954,16-Oct-2001
VX8161,44641,4-May-2009

IX8371,76522,7-May-2009

Insert into located_at (aid, pubid) ->

aid,pubid
1032,37270
1033,45671
1034,27542
1035,87623
1036,12076
1037,31677
1038,17588
1039,15470
1040,77523
1041,73864
1042,32585
1043,79566
1044,32527
1045,75892
1046,85680
1047,451
1048,45762
1049,73243
1050,28954
1051,44641
1052,76522
1053,14891
1054,98567
1055,25228
1056,45629
1057,12354
1058,3485

Insert into lives_at (aid, pid) ->

aid,pid
1000,102
1001,103
1002,106
1003,107
1004,108
1005,117
1006,118
1007,119
1008,120
1009,121

1010,122
1011,123
1012,124
1013,125
1014,126
1015,29
1016,131
1017,134
1018,135
1019,141
1020,142
1021,145
1022,146
1023,147
1024,150
1025,151
1026,152
1027,153
1028,200
1029,201
1030,202
1031,203

Insert into membership (memberid, length, start_date, fee) ->

memberid,length,start_date,fee
5bfb0970fc13ae4b54000000,12,11/19/19,500
5bfb0970fc13ae4b54000001,23,04/19/19,250
5bfb0970fc13ae4b54000002,12,09/23/23,300
5bfb0970fc13ae4b54000003,3,06/10/09,50
5bfb0970fc13ae4b54000004,2,05/24/24,60
5bfb0970fc13ae4b54000005,8,01/21/20,70
5bfb0970fc13ae4b54000006,18,01/05/04,80
5bfb0970fc13ae4b54000007,12,09/05/06,90
5bfb0970fc13ae4b54000008,8,08/31/53,100
5bfb0970fc13ae4b54000009,12,10/27/17,300

Insert into join (pid, mid) ->

pid,mid
102,5bfb0970fc13ae4b54000000
108,5bfb0970fc13ae4b54000001
117,5bfb0970fc13ae4b54000002
118,5bfb0970fc13ae4b54000003
120,5bfb0970fc13ae4b54000004

121,5bfb0970fc13ae4b54000005
124,5bfb0970fc13ae4b54000006
125,5bfb0970fc13ae4b54000007
126,5bfb0970fc13ae4b54000008
145,5bfb0970fc13ae4b54000009

Insert into member_search (memberid, eid, retrived) ->

memberid,eid,retrieved
5bfb0970fc13ae4b54000000,IX3646,1/10/2019
5bfb0970fc13ae4b54000001,IB2946,1/11/2019
5bfb0970fc13ae4b54000002,LH8542,1/12/2019
5bfb0970fc13ae4b54000003,LH8542,1/13/2019
5bfb0970fc13ae4b54000004,LH8542,1/14/2019
5bfb0970fc13ae4b54000005,LH8542,1/15/2019
5bfb0970fc13ae4b54000006,LH8542,1/16/2019
5bfb0970fc13ae4b54000007,UE6668,1/17/2019
5bfb0970fc13ae4b54000008,RY2195,1/18/2019
5bfb0970fc13ae4b54000009,KQ4175,1/19/2019

Insert into nonmember_search(pid, eid, fee, search_date) ->

pid,eid,fee,search_date
141,IX3646,0.1,1/10/2019
142,IB2946,0.2,2/11/2019
145,LH8542,0.3,3/15/2019
146,YZ6317,0.4,4/16/2019
147,AB2755,0.5,5/18/2019
150,OV3250,0.6,6/19/2019
151,ZL9928,0.7,7/21/2019
152,UE6668,0.8,8/22/2019
153,RY2195,0.9,9/23/2019
200,KQ4175,1,10/25/2019
201,VX8161,1.1,11/26/2019
202,IX8371,1.2,12/28/2019
203,VX8161,1.3,1/29/2020

Basic interactions and Additional Queries

Queries

SQL query 1:

Description:

Get the first name, last name, and user name of everyone who lives in minnesota.

SQL:

```
select p.fname, p.lname, u.uname
from person p, lives_at l, address a, users u
where p.pid = l.pid AND
        l.aid = a.aid AND
        p.pid = u.pid AND
        a.state = 'Minnesota';
```

Output:

	FNAME	LNAME	UNAME
1	Korie	Richemont	mhillandd
2	Tim	Jeffreys	ggarlingej

Justification:

The username from the USERS table is found by equating the pid from the PERSON and USERS table. The LIVES_AT table is used to connect a person's pid with an addresses aid. The line "a.state = 'Minnesota'" ensures that only persons living in Minnesota are found.

SQL query 2:

Description:

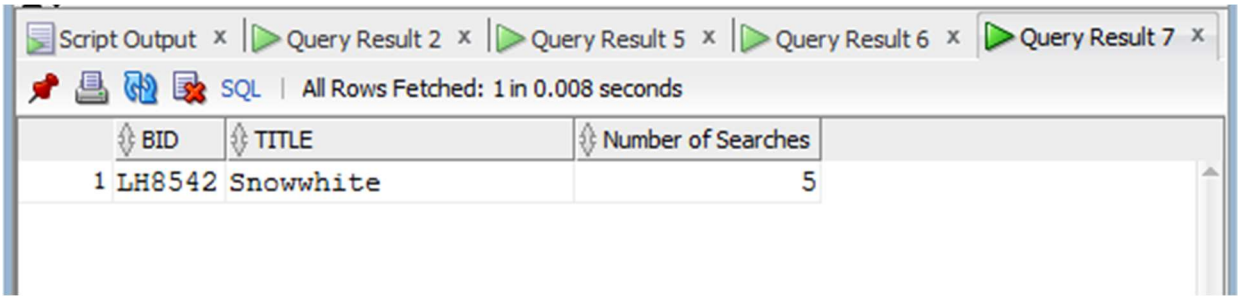
Most searched item in the library

SQL:

```
SELECT B.bid, E.title, temp.num as "Number of Searches"
FROM Entry E, Book B, (SELECT *
                FROM (
                SELECT S.eid, COUNT(S.eid) as num
                FROM MEMBER_SEARCH S
                GROUP BY S.eid
                ORDER BY COUNT(S.eid) DESC )
                WHERE ROWNUM = 1 ) temp
WHERE temp.eid = B.bid AND E.eid = B.bid
UNION
```

```
SELECT M.mid, E.title, temp.num as "Number of Searches"
From Entry E, Magazine M, (SELECT *
                        FROM (
                        SELECT S.eid, COUNT(S.eid) as num
                        FROM MEMBER_SEARCH S
                        Group BY S.eid
                        ORDER BY COUNT (S.eid) DESC )
                        WHERE ROWNUM = 1) temp
WHERE temp.eid = M.mid AND E.eid = M.mid
UNION
SELECT J.jid, E.title, temp.num as "Number of Searches"
From Entry E, Journal J, (SELECT *
                        FROM (
                        SELECT S.eid, COUNT(S.eid) as num
                        FROM MEMBER_SEARCH S
                        Group BY S.eid
                        ORDER BY COUNT (S.eid) DESC )
                        WHERE ROWNUM = 1) temp
WHERE temp.eid = J.jid AND E.eid = J.jid
UNION
SELECT C.cid, E.title, temp.num as "Number of Searches"
From Entry E, CONFERENCE_PROCEEDINGS C, (SELECT *
                        FROM (
                        SELECT S.eid, COUNT(S.eid) as num
                        FROM MEMBER_SEARCH S
                        Group BY S.eid
                        ORDER BY COUNT (S.eid) DESC )
                        WHERE ROWNUM = 1) temp
WHERE temp.eid = C.cid AND E.eid = C.cid;
```

Output:



	BID	TITLE	Number of Searches
1	LH8542	Snowwhite	5

Justification:

The four entry types (Journal, Magazine, Book, and Conference Proceedings) are grouped by entryID and a count is taken. The query selects the item with the highest count, which is the most number of searches.

SQL query 3:

Description:

2) Names, emails and fee paid by each member

SQL:

```
SELECT p.fname, p.lname, p.email, m.fee
FROM Login x, Person p, Membership m
WHERE x.pid = p.pid AND x.MEMBERID = m.MEMBERID;
```

Output:

	FNAME	LNAME	EMAIL	FEE
1	Reiko	Brettle	rbrettle9@1688.com	500
2	Morton	Northern	mnortherna@so-net.ne.jp	250
3	Aubrey	Lacroutz	alacroutzb@sbwire.com	300
4	Paton	Tuxwell	ptuxwellc@eepurl.com	50
5	Emmey	Spearman	espearmand@livejournal.com	60
6	Shir	Hovell	shovelle@angelfire.com	70
7	Thaxter	Winston	twinstonf@istockphoto.com	80
8	Andee	Lecount	alecountg@unesco.org	90
9	Kelley	Lewisham	klewishamh@edublogs.org	100
10	Saxon	Garrard	sgarrardi@dell.com	300

Justification:

All the person IDs from login table are matched with person IDs from person table and then member IDs from login table are matched with member IDs from membership table so that output shows the first name, last name , email and membership fee paid by each person.

SQL query 4:

Description:

Find the First name, Last name and email address for the author with the most authored pieces.

SQL and Output:

```
SELECT * FROM (
    SELECT x.fname, x.lname, x.email, a.pid
    FROM Person x, Authored a
    WHERE a.pid = x.pid
    GROUP BY (x.fname, x.lname, x.email, a.pid)
    ORDER BY COUNT(x.pid) DESC
   )
   WHERE ROWNUM=1;
```

Script Output × | Query Result ×

SQL | All Rows Fetched: 1 in 0.029 seconds

	FNAME	LNAME	EMAIL	PID
1	Archibaldo	Warnes	awarnes3@buzzfeed.com	107

Justification:

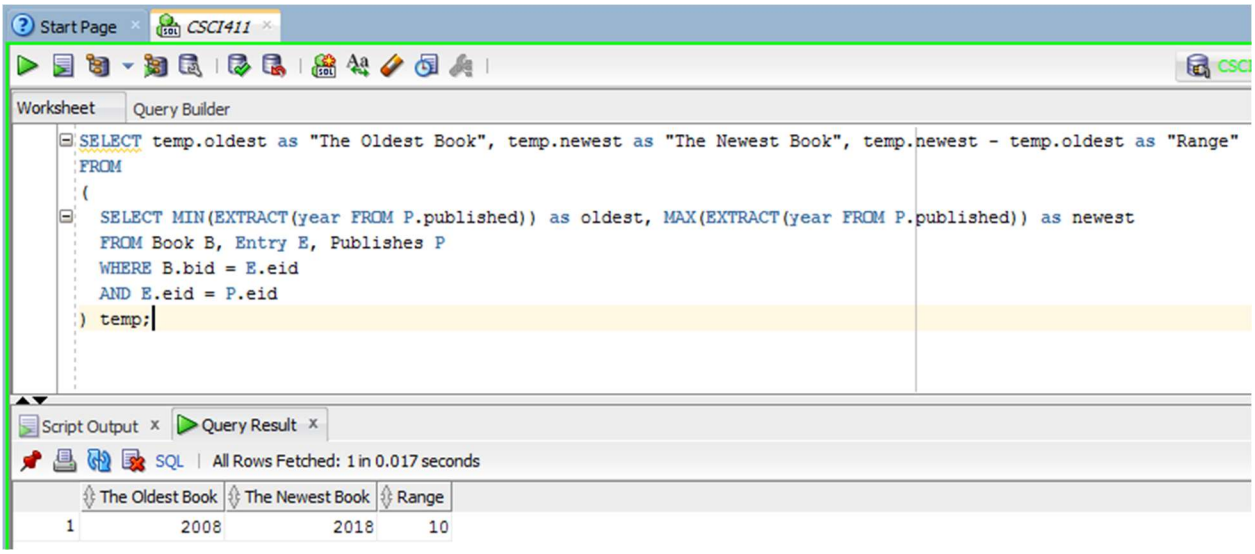
Finds list of authors with the most authored pieces in descending order and chooses the first one from the list.

SQL query 5:

Description:

Give the age range between the oldest and newest book in the library.

SQL and Output:



Justification:

The oldest and newest books are selected by using equijoin between bid and eid.The range is calculated by extracting the date when the newest and oldest book were published and subtracting their values.

SQL query 6:

Description:

The publishing company with the most books published.

SQL and Output:

```
-- The publishing company with the most books
SELECT *
FROM            (SELECT PC.pubname, PC.pubid, PC.email
                 FROM PUBLISHER PC, PUBLISHES P, Book B
                 WHERE P.eid = B.bid
                 AND PC.pubid = P.pubid
                 GROUP BY (PC.pubname, PC.pubid, PC.email)
                 ORDER BY COUNT(B.bid) DESC)
WHERE ROWNUM = 1;
```

All Rows Fetched: 1 in 0.018 seconds

	PUBNAME	PUBID	EMAIL
1	Mayer-Fadel	27542	mf@mail.net

Justification:

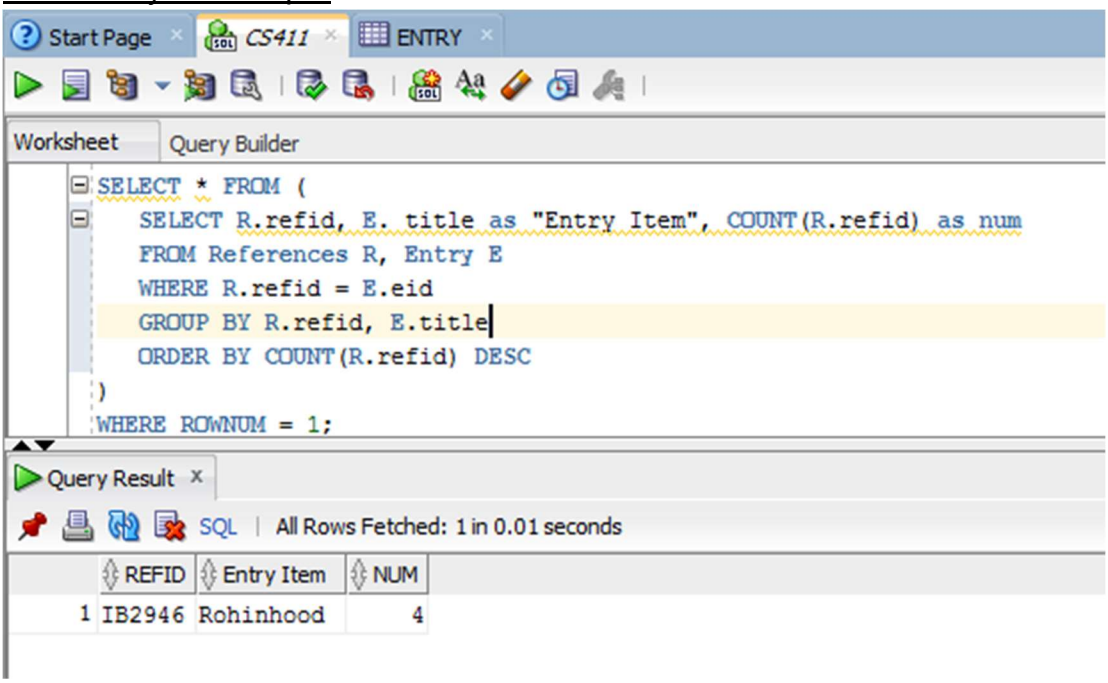
The Publishing Companies who published books are ordered by the highest count of books. The Publishing Company with the highest count of published books is selected. The query displays the Publishing Company's name, id and email.

SQL Query 7:

Description:

Find the entry item's reference ID and name for the item which has been referenced the most.

SQL Query and Output



Justification:

Orders the list of items referenced in a descending order and chooses the first tuple from the table.

SQL query 8:

Description:

Get the titles of entries in the library that have been searched for by non-members but not members.

SQL:

```
select e.title
from entry e, non_member_search n
where e.eid = n.eid
minus
select e.title
```

```
from entry e, member_search ms
where e.eid = ms.eid;
```

Output:

	TITLE
1	American Bankers Association
2	American Computer Scientist
3	American Medical Soceity
4	Computer Vision for Heart surgery
5	Deep Learning for Surgery
6	Forbes

Justification:

The title of items in the library is found for both the member and non-member searches, the titles of books that members searched for is subtracted from the titles of books that non-members searched for.

SQL query 9:

Description:

Get the dates where both members and non-members have searched for something.

SQL:

```
select ms.retrieved
from users u, login l, membership m, member_search ms
where u.pid = l.pid AND
        l.memberid = m.memberid AND
        m.memberid = ms.memberid
intersect
select ns.search_date
from users u, non_member_search ns
where u.pid = ns.pid;
```

Output:

	RETRIEVED
1	10-JAN-19

Justification:

The dates of searches in the NON_MEMBER_SEARCH and MEMBER_SEARCH tables are found, and an intersection is used to find where they are in common. January 10th, 2019 is the only date in both tables.

SQL query 10:

Description:

Get the emails of all the authors who have authored something in the library.

SQL:

```
select distinct p.email
from person p, author a, authored au
where p.pid = a.pid AND
        a.pid = au.pid;
```

Output:

	EMAIL
1	jdysert7@reddit.com
2	asandcraft8@ucla.edu
3	awarnes3@buzzfeed.com
4	smeriel0@nih.gov
5	mgrayston4@ezinearticles.com
6	pfinley1@domainmarket.com
7	halleburton2@vinaora.com
8	tgear5@yelp.com
9	cdohmer6@hhs.gov

Justification:

The emails from the PERSON table are found with the select statement. If a pid is found in the AUTHORED table, then the associated email is output.

Stored procedures

Stored procedure 1:

Description:

Add new users to the USERS table.

SQL:

```
create or replace procedure adduser(pid in number, uname in varchar2, fname in
varchar2, lname in varchar2, email in varchar2) AS
begin
  insert into PERSON values(pid, fname, lname, email);
  insert into USERS values (pid, uname);
end;
```

Output:

```
exec adduser(999, 'newuser', 'firstname', 'lastname', 'newuser@gmail.com');
```

	PID	UNAME
1	121	kcard0
2	122	jcisar1
3	123	smccaskill2
4	124	gpoulney3
5	125	anewlove4
6	126	pconnar5
7	29	cofogerty6
8	131	edurrell7
9	134	mverrico8
10	135	etheze9
11	141	nspira
12	142	bminnisb
13	145	ksymsonc
14	146	mhillandd
15	147	sdrapere
16	150	tannesleyf
17	151	bchalcotg
18	152	dmeuseh
19	153	wsherringtoni
20	200	ggarlingej
21	201	blugsdink
22	202	lgopsalll
23	203	gbaudassim
24	555	jdoe
25	999	newuser

Justification:

The table in the output above shows the addition of a new user with uname of ‘newuser’ and pid of 999 into the USERS table.
A new entry in the PERSON table is created first to allow for the creation of a new user in the USERS table. The new user shows up in the USERS table and the PERSON table.

Stored procedure 2:

Description:

Delete a user from the USERS table.

SQL:

create or replace procedure deleteuser(deluser in varchar2) as

```
begin
delete from users where uname = deluser;
end;
```

Output:

```
exec deleteuser('newuser');
```

	PID	UNAME
1	121	kcard0
2	122	jcisar1
3	123	smccaskill2
4	124	gpoulney3
5	125	anewlove4
6	126	pconnar5
7	29	cofogerty6
8	131	edurrell7
9	134	mverrico8
10	135	etheze9
11	141	nspira
12	142	bminnisb
13	145	ksymsonc
14	146	mhillandd
15	147	sdrapere
16	150	tannesleyf
17	151	bchalcotg
18	152	dmeuseh
19	153	wsherringtoni
20	200	ggarlingej
21	201	blugsdink
22	202	lgopsalll
23	203	gbaudassim
24	555	jdoe

Justification:

The table in the output above shows the removal of a user, from the USERS table, with the uname of ‘newuser’. This user was added during the demonstration of the previous stored procedure.
A delete statement is used to remove the record from the USERS table. The associated person record remains in the database as desired.

Stored procedure 3:

Description:

Non-member search of the digital library.

SQL:

```
create or replace procedure nonmembersearch(searchedfor in varchar2, username in
varchar2, searchdate in date default SYSTIMESTAMP, fee in float default .05) as

  curs varchar2(255);
  pid number;
  eid varchar2(255);

  cursor ref_out is
  select e.body
  from entry e
  where e.title = searchedfor;

begin
select u.pid into pid from users u where u.uname = 'blugsdink';
select e.eid into eid from entry e where e.title = 'Snowwhite';

  insert into non_member_search values(pid, eid, searchdate, fee);

  open ref_out;
        fetch ref_out into curs;
        dbms_output.put_line(searchedfor || ', ' || curs);
  close ref_out;

end;
```

Output:

```
set serveroutput on;
exec nonmembersearch('Snowwhite', 'blugsdink');
```

```
PL/SQL procedure successfully completed.

Snowwhite, The content of the book is tales of Snowwhite
```

	PID	EID	SEARCH_DATE	FEE
1	141	IX3646	10-JAN-19	0.1
2	142	IB2946	11-FEB-19	0.2
3	145	LH8542	15-MAR-19	0.3
4	146	YZ6317	16-APR-19	0.4
5	147	AB2755	18-MAY-19	0.5
6	150	OV3250	19-JUN-19	0.6
7	151	ZL9928	21-JUL-19	0.7
8	152	UE6668	22-AUG-19	0.8
9	153	RY2195	23-SEP-19	0.9
10	200	KQ4175	25-OCT-19	1
11	201	VX8161	26-NOV-19	1.1
12	202	IX8371	28-DEC-19	1.2
13	203	VX8161	29-JAN-20	1.3
14	201	LH8542	28-NOV-18	0.05

Justification:

The table in the output shows that a new search has been recorded in the NON_MEMBER_SEARCH table on the date of November 28, 2018.
A new entry is created to the NON_MEMBER_SEARCH table for the given user and entry title that is searched for; importantly, the time of the search is recorded and a default fee is charged for the search. Select statements are used to get the pid and eid of the user and the entry's title.

Stored procedure 4:

Description:

Who accessed _____?

SQL:

```
create or replace procedure whoaccessed(searchedfor in varchar2) as
  curs varchar2(255);

  cursor ref_out is
  select u.uname
  from USERS u, Entry e, NON_MEMBER_SEARCH n
  where e.title = searchedfor AND
        u.pid = n.pid AND
        n.eid = e.eid
  union
```

```
  select u.uname
  from Users u, Entry e, Login l, membership m, MEMBER_SEARCH ms
  where e.title = searchedfor AND
        u.pid = l.pid AND
        l.memberid = m.memberid AND
        m.memberid = ms.memberid AND
        ms.eid = e.eid;
begin
        open ref_out;
        loop
        fetch ref_out into curs;
        if ref_out %NOTFOUND then exit; end if;
        dbms_output.put_line(curs);
        end loop;
end;
```

Output:

```
set serveroutput on;
exec whoaccessed('Snowwhite');
```

```
PL/SQL procedure successfully completed.

anewlove4
blugsdink
cofogerty6
gpoulney3
ksymsonc
pconnar5
smccaskill2
```

Justification:

This stored procedure uses a union to combine the results of the non-member search with the results of the member search.
In the NON_MEMBER_SEARCH table there is one entry with an eid of ‘LH8542’ and in the MEMBER_SEARCH table there are five searches with that eid, for a total of six entries.
The output shows six usernames associated with each of the six entries as desired.

Stored procedure 5:

Description:

What articles references article x?

SQL:

```
create or replace procedure articlesThatReference(X in VARCHAR2) as
  articlesThatRef VARCHAR(100);
  CURSOR   articlesCursor is
  SELECT   E2.title
  FROM        REFERENCES R, REFERENCES R2, Entry E, Entry E2
  WHERE       E.title = X       AND
          R.eid = E.eid         AND
          R2.refid = R.eid      AND
          R2.eid = E2.eid;
begin
          open articlesCursor;
          loop
          fetch articlesCursor into articlesThatRef;
          if articlesCursor %NOTFOUND then exit; end if;
          dbms_output.put_line('Article named: ' || X ||
          ' is referenced by the article: ' || articlesThatRef);
          end loop;
          close articlesCursor;
end;
set serveroutput on;
EXEC articlesThatReference('Snowwhite');
```

Output:

Script Output x Query Result x

Task completed in 0.072 seconds

```
Procedure ARTICLESTHATREFERENCE compiled

PL/SQL procedure successfully completed.

Article named: Snowwhite is referenced by the article: Rohinhood
Article named: Snowwhite is referenced by the article: Success
```

Justification:

An article name is input. The reference table is scanned for articles that reference the user input article.

Stored procedure 6:

Description:

What articles does article x reference?

SQL and Output:

```
-- What articles does article x reference?
create or replace procedure articleXreferences (X in VARCHAR2) as
articleReferenced VARCHAR2(100);
  begin
  SELECT   E2.title into articleReferenced
  FROM     REFERENCES R, Entry E, Entry E2
  WHERE    E.title = X          AND
           R.eid = E.eid        AND
           E2.eid = R.refid;
          dbms_output.put_line('Article: ' || X || ' references article: '
          || articleReferenced);
end;
set serveroutput on;
EXEC articleXreferences('Snowwhite');
```

```
Procedure ARTICLEXREFERENCES compiled

PL/SQL procedure successfully completed.

Article: Snowwhite references article: American Bankers Association
```

Justification:

The user input entry title is compared with the reference table to find the articles it references. The referenced article is compared with the entry table to get the title.

Stored procedure 7:

Description:

Get author X's email address and postal address.

SQL and Output:

```
-- 1.Author x's email address and postal address
create or replace procedure getAuthorEmailandAddress (firstname in VARCHAR2, lastname in VARCHAR2) as
email VARCHAR2(80);
st VARCHAR2(50);
city VARCHAR(30);
state VARCHAR(20);
  begin
    SELECT  P.email, Ad.street,Ad.city, Ad.state into email, st,city, state
    FROM    AUTHOR A, Person P, Lives_At L, Address Ad
    WHERE   P.fname = firstname   AND
            P.lname = lastname    AND
            A.pid = P.pid         AND
            L.pid = P.pid         AND
            L.aid = Ad.aid;
            dbms_output.put_line('Author: '  || firstname || ' ' || lastname || CHR(10) ||
                                 'Email Address: ' || email || CHR(10) ||
                                 'Postal Address: ' || CHR(10) ||
                                 st || CHR(10) ||
                                 city || ', ' || state ||  '.');
end;
set serveroutput on;
EXEC getAuthorEmailandAddress('Serge', 'Meriel');
```

```
Procedure GETAUTHOREMAILANDADDRESS compiled

PL/SQL procedure successfully completed.

Author: Serge Meriel
Email Address: smeriel0@nih.gov
Postal Address:
7671 Marcy Point
Oceanside, California.
```

Justification:

The author's first name and last name must be input to retrieve the author's email address and postal address. Variables were created to store the needed selected author information. The stored procedure verifies that the first name and last name are that of an author. Once the author is verified, the name, email address, and postal address are displayed.

Stored procedure 8:

Description:

Register a new member.

SQL:

```
create or replace procedure registernewmember(pid in number, memberid in varchar2, fname in varchar2, lname in varchar2, email in varchar2, uname varchar2, startdate in date default SYSTIMESTAMP, length in number default 31, fee in float default 12.9) as
begin
  insert into person values(pid, fname, lname, email);
  insert into users values(pid, uname);
  insert into membership values(memberid, length, startdate, fee);
  insert into login values(pid, memberid);
end;
```

Output:

```
exec registernewmember(555, 'ad3qaf34af', 'John', 'Doe', 'jdoe@gmail.com', 'jdoe');
```

	MEMBERID	LENGTH	START_DATE	FEE
1	5bfb0970fc13ae4b54000000	12	19-NOV-19	500
2	5bfb0970fc13ae4b54000001	23	19-APR-19	250
3	5bfb0970fc13ae4b54000002	12	23-SEP-23	300
4	5bfb0970fc13ae4b54000003	3	10-JUN-09	50
5	5bfb0970fc13ae4b54000004	2	24-MAY-24	60
6	5bfb0970fc13ae4b54000005	8	21-JAN-20	70
7	5bfb0970fc13ae4b54000006	18	05-JAN-04	80
8	5bfb0970fc13ae4b54000007	12	05-SEP-06	90
9	5bfb0970fc13ae4b54000008	8	31-AUG-53	100
10	5bfb0970fc13ae4b54000009	12	27-OCT-17	300
11	ad3qaf34af	31	27-NOV-18	12.9

Justification:

The table in the output shows the addition of a new entry into the MEMBER table with the memberid of ‘ad3qaf34af’ as desired.
Registering a new member involves creating a new entry in the PERSON table, a new user in the USERS table, and a new member in the MEMBERSHIP table. As well as adding the memberid and pid from the MEMBERSHIP table and the USERS table into the login table. Insert statements are used to add these new entries, the cost and length of the membership have default values.

Stored procedure 9:

Description:

Remove a user’s membership, keeping the user account in place.

SQL:

```
create or replace procedure deletemember(username in varchar2) as

  pid number;
  memid varchar2(255);
begin
  select u.pid into pid from users u where u.uname = username;
  select l.memberid into memid from login l, users u where u.uname = username AND
u.pid = l.pid;

  delete from member_search where memberid = memid;
  delete from login where memberid = memid;
 delete from membership where memberid = memid;
end;
```

Output:

```
exec deletemember('cofogerty6');
```

	MEMBERID	LENGTH	START_DATE	FEE
1	5bfb0970fc13ae4b54000000	12	19-NOV-19	500
2	5bfb0970fc13ae4b54000001	23	19-APR-19	250
3	5bfb0970fc13ae4b54000002	12	23-SEP-23	300
4	5bfb0970fc13ae4b54000003	3	10-JUN-09	50
5	5bfb0970fc13ae4b54000004	2	24-MAY-24	60
6	5bfb0970fc13ae4b54000005	8	21-JAN-20	70
7	5bfb0970fc13ae4b54000007	12	05-SEP-06	90
8	5bfb0970fc13ae4b54000008	8	31-AUG-53	100
9	5bfb0970fc13ae4b54000009	12	27-OCT-17	300

Justification:

The table in the output shows the removal of the entry with memberid of ‘5bfb0970fc13ae4b540000006’ as desired.
The foreign key in the MEMBERSHIP table does not have a cascading delete so the entries in the LOGIN and MEMBER_SEARCH tables must be deleted first. The USERS table is needed to make it possible to delete the entry in the MEMBERSHIP table without knowing the memberid.

Stored procedure 10:

Description:

Add a book to the library.

SQL:

```
create or replace procedure addbook(bid in varchar2, booktitle in varchar2, bookbody in varchar2, numpages in number, bedition in number default 1) as

begin
  insert into Entry(eid, title, body) values(bid, booktitle, bookbody);
  insert into book values(bid, bedition, numpages);
end;
```

Output:

```
exec addbook('HH3719', 'The Newest Book', 'This is the content of The Newest Book', 128);
```

	BID	BEDITION	NUMPAGES
1	IX3646	1	900
2	IB2946	2	200
3	LH8542	1	750
4	NJ6977	3	600
5	CE7059	5	200
6	VM7843	8	500
7	HH3719	1	128

Justification:

The table in the output shows the addition of ‘HH3719’ to the BOOK table.
A new entry in the Entry table must be made before the book can be made since a book is an entry. Both additions are made with the insert statement.