Contents

Introduction

The aim of this report is to implement the College Library System database designed in the first coursework. This will be accomplished by creating tables, creating test data for those tables, inserting the data into the tables, creating views, creating SQL queries to test the database and creating triggers with demonstrations.

The database is intended to automate library activities such as creating new members, allowing them to borrow resources and maintaining the details of all the items that are available in the resources. This also helps the library staff by providing information such as the number of copies for each resource and how long the resource can be borrowed for.

Relational Schema

Relational Schema from coursework 1:
Member (Member_ID, Member_Type, Member_Name, Member_Email, Expiry_Date)
Resource (Resource_ID, Resource_Type, Resource_Title, Total_Copies, Loan_Period, Floor_No, Shelf_No, Author, Publisher, Publish_Date, Edition, Director, Release_Date, Artist)
Loan (Loan_ID, Return_Date, Date_Paid, Amount_Paid)
Class (Class_No, Class_Name)
Borrow (Member_ID, Resource_ID, Due_Date, Loan_ID)
Enrol (Member_ID, Class_No, Enrol_Date)
Copies (Member_Type, Copies_Allowed)

Some adjustments were made to the relational schema created in the previous coursework: Class_No should have been in the Res table but it was missing in the normalised relations in coursework 1. The publisher, director and artist attributes in the Res table have been combined into one attribute (Publsh_Drct_Art). This was done as the value of only one attribute is NOT NULL for a resource. The Release_Date and Publish_Date attributes in the Res table have been combined into one attribute (Release_Date) for the same reason. The Loan table was split into the Return and Fine_Payment tables to avoid NULL values. This has been done as not every return will contain a fine.

Revised Relational Schema:
Copies (Member_Type, Copies_Allowed)
MEMBER (Member_ID, *Member_Type*, Member_Name, Member_Email, Expiry_Date)
Class (Class_No, Class_Name)
Res (Resource_ID, Resource_Title, Total_Copies, Loan_Period, Floor_No, Shelf_No, *Class_No*, Resource_Type, Author, Release_Date, Publsh_Drct_Art, Edit)
Borrow (*Member_ID*, *Resource_ID*, Due_Date, Loan_ID)
Return (*Loan_ID*, Return_Date)
Fine_Payment (*Loan_ID*, Date_Paid, Amount_Paid)
Enrol (*Member_ID*, *Class_No*, Enrol_Date)

Create Tables

Copies:
The Copies table has Member_Type as the primary key constraint. This declares that the value for this column must be unique as well as NOT NULL. The data type of Member_Type is denoted by a string with a character limit of 20. The other columns in the table are named Member_Type and Copies_Allowed. number(10) is the other data type. The Member_Type attribute contains a check constraint, meaning that its value must be either Student or Staff. The Copies_Allowed attribute also contains a check constraint, meaning that its value must be either 5 or 10.

```
CREATE TABLE Copies
( Member_Type varchar2(20),
  Copies_Allowed number(10),
  CONSTRAINT Member_Type
   CHECK (Member_Type IN ('Student', 'Staff')),
  CONSTRAINT Copies_Allowed
   CHECK (Copies_Allowed IN (5, 10)),
  CONSTRAINT Copies_pk PRIMARY KEY (Member_Type)
);
```

MEMBER:
The MEMBER table has Member_ID as the primary key constraint. The data type of Member_ID is denoted by the number 30, which means that the values entered must not exceed 30 digits. The other columns in the table are named Member_Type, Member_Name, Member_Email and Expiry_Date. The only data types in the columns are string and DATE. The email attributes contains two checks which will verify that the string contains '@' as well as '.'. The Member_Type, Member_Name, Member_Email and Expiry_Date attributes are considered essential for the MEMBER table and therefore contain the NOT NULL constraint. The Member_Type attribute referenced from the Copies table contains a foreign key constraint. This creates a link between the two tables and enforces referential integrity by ensuring that values that are not found in the Copies table are not entered.

```
CREATE TABLE MEMBER
( Member_ID number(30),
  Member_Type varchar2(20) NOT NULL,
  Member_Name varchar2(250) NOT NULL,
  Member_Email varchar2(1000) NOT NULL UNIQUE,
  Expiry_Date DATE NOT NULL,
  CHECK (Member_Email LIKE '%@%'),
  CHECK (Member_Email LIKE '%.%'),
  CONSTRAINT Mem_pk PRIMARY KEY (Member_ID),
  CONSTRAINT Mem_fk FOREIGN KEY (Member_Type) REFERENCES Copies(Member_Type)
);
```

Class:
The Class table has Class_No as the primary key constraint. Class_No is automatically generated in a sequential order starting from 101. The other column in the table is named Class_Name which is a string data type.

```
CREATE TABLE Class
( Class_No number GENERATED ALWAYS AS IDENTITY start with 101,
  Class_Name varchar2(250) NOT NULL UNIQUE,
  CONSTRAINT Class_pk PRIMARY KEY (Class_No)
);
```

Res:
The Res table has Resource_ID as the primary key constraint. Resource_ID is automatically generated in a sequential order starting from 100001. The other columns in the table are named Resource_Title, Total_Copies, Loan_Period, Floor_No, Shelf_No, Class_no, Resource_Type, Author, Release_Date, Publsh_Drct_Art and Edit. The Loan_Period attribute contains a check constraint meaning that its value must be either 0, 2 or 14. The Floor_No attribute also contains a check constraint meaning that its value must be either 0, 1 or 2. The Resource_Type attribute contains a check constraint meaning that its value must be either Book, Video, CD or DVD. The Class_No attribute referenced from the Class table has a foreign key constraint. The Floor_No and Shelf_No attributes contain a unique constraint as no two separate resources can be stored at the same location.

```
CREATE TABLE Res
(Resource_ID NUMBER GENERATED ALWAYS AS IDENTITY start with 100001,
  Resource_Title varchar2(250) NOT NULL,
  Total_Copies number(10) NOT NULL,
  Loan_Period number(10) NOT NULL,
  Floor_No number(10) NOT NULL,
  Shelf_No varchar2(20) NOT NULL,
  Class_no number(10) NOT NULL,
  Resource_Type varchar2(5) NOT NULL,
  Author varchar2(250),
  Release_Date Date NOT NULL,
  Publsh_Drct_Art varchar2(250) NOT NULL,
  Edit Number,
  CONSTRAINT Loan_Period_Check
   CHECK (Loan_Period IN (0, 2, 14)),
  CONSTRAINT Floor_No_check
   CHECK (Floor_No IN (0,1,2)),
   CHECK (Resource_Type IN ('Book', 'Video', 'CD', 'DVD')),
  UNIQUE(Floor_No, Shelf_No),
  CONSTRAINT Res_pk PRIMARY KEY (Resource_ID),
  CONSTRAINT Res_fk FOREIGN KEY (Class_No) REFERENCES Class(Class_No)
);
```

Borrow:
The Borrow table has Member_ID, Resource_ID and Due_Date as the primary key constraints. The other column in the table is named Loan_ID. Loan_ID is automatically generated in a sequential order starting from 1000001. The attributes Member_ID and Resource_ID referenced from the MEMBER and Res tables respectively, both contain a foreign key constraint.

```
CREATE TABLE Borrow
( Member_ID number(30),
  Resource_ID NUMBER,
  Due_Date DATE NOT NULL,
  Loan_ID NUMBER GENERATED ALWAYS AS IDENTITY start with 1000001 UNIQUE,
  CONSTRAINT Borrow_pk PRIMARY KEY (Member_ID, Resource_ID, Due_Date),
  CONSTRAINT Borrow_fk1 FOREIGN KEY (Member_ID) REFERENCES MEMBER(Member_ID),
  CONSTRAINT Borrow_fk2 FOREIGN KEY (Resource_ID) REFERENCES Res(Resource_ID)
);
```

Return:
The Return table has Loan_ID as the primary key constraint. The other column in the table is named Return_Date. The Loan_ID attribute referenced from the Borrow table also contains a foreign key constraint.

```
CREATE TABLE Return
( Loan_ID NUMBER,
  Return_Date DATE NOT NULL,
  CONSTRAINT Return_pk PRIMARY KEY (Loan_ID),
  CONSTRAINT Return_fk FOREIGN KEY (Loan_ID) REFERENCES Borrow(Loan_ID)
);
```

Fine_Payment:
The Fine_Payment table has Loan_ID as the primary key constraint. The other columns in the table are named Date_Paid and Amount_Paid. The Loan_ID attribute referenced from the Borrow table also contains a foreign key constraint.

```
CREATE TABLE Fine_Payment
( Loan_ID NUMBER,
  Date_Paid Date NOT NULL,
  Amount_Paid number(20) NOT NULL,
  CONSTRAINT FinePayment_pk PRIMARY KEY (Loan_ID),
  CONSTRAINT FinePayment_fk FOREIGN KEY (Loan_ID) REFERENCES Borrow(Loan_ID)
);
```

Enrol:
The Enrol table has Member_ID and Class_No as the primary key constraints. The other column in the table is named Enrol_Date. The attributes Member_ID and Class_No referenced from the MEMBER and Class tables respectively, also contain foreign key constraints.

```
CREATE TABLE Enrol
( Member_ID number(30),
  Class_No NUMBER,
  Enrol_Date DATE NOT NULL,
  CONSTRAINT Enrol_pk PRIMARY KEY (Member_ID, Class_No),
  CONSTRAINT Enrol_fk1 FOREIGN KEY (Member_ID) REFERENCES MEMBER(Member_ID),
  CONSTRAINT Enrol_fk2 FOREIGN KEY (Class_No) REFERENCES Class(Class_No)
);
```

Data

The following section contains the data which was inserted in the tables. Some values in the code were deliberately set to fail and therefore check the validation. The code has been listed as well as the reason for giving errors.

Copies:

Member_Type	Copies_Allowed
Student	5
Staff	10

The Copies table has no dependency to other tables.

INSERT INTO Copies
(Member_Type, Copies_Allowed)
VALUES ('Employer', 5)
This cannot be inserted as the value of Member_Type can only be Student or Staff, not Employer.

INSERT INTO Copies
(Member_Type, Copies_Allowed)
VALUES ('Student', 12)
This cannot be inserted as the value of Copies_Allowed can only be 5 or 10, not 12.

INSERT INTO Copies
(Copies_Allowed)
VALUES (7)
This cannot be inserted as the primary key cannot be NULL.

MEMBER:

Member_ID	Member_Type	Member_Name	Member_Email	Expiry_Date
202001	Student	Shajeda	shajedasyed@yahoo.com	23-Jan-23
202002	Student	Kayley	kanett@hotmail.co.uk	28-Jan-24
202003	Student	Purvi	purvimandodadoda@hotmail.co.uk	21-Jan-23
202004	Student	Steven	stevencorpuz@gmail.co.uk	14-Jan-23
202005	Student	Hana	hanarahman@hotmail.com	29-Jan-21
202006	Student	Rohit	rohitgarg12@hotmail.com	14-Jan-23
202007	Student	Tanya	tanyasyed_@yahoo.com	20-Jan-24
202008	Student	Cef	cefalrahman@hotmail.com	22-Jan-24
202009	Student	Daniel	debenham_dan@hotmail.co.uk	25-Jan-23
202010	Student	Holly	hollylou@outlook.co.uk	30-Jan-23
202011	Student	Rafayet	rafayettarafder@live.co.uk	21-Jan-22
202012	Student	Ope	oppeeeee@outlook.co.uk	21-Jan-22
202013	Student	Jack	jack_1995@hotmail.co.uk	25-Jan-23

202014	Student	Teddy	teddy.moonshine@live.co.uk	29-Jan-22
202015	Student	Maria	maria.678@gmail.com	21-Jan-21
202101	Staff	Vicky	v.anderson@qmul.ac.uk	25-Jan-29
202102	Staff	Mahesha	m.soong@qmul.ac.uk	25-Jan-31
202103	Staff	Chris	c.davidson2@qmul.ac.uk	25-Jan-35
202104	Staff	Fabrizio	c.caine@qmul.ac.uk	19-Jan-21
202105	Staff	Mark	m.amaral@qmul.ac.uk	28-Jan-33
202106	Staff	Sophie	s.willoughby@qmul.ac.uk	29-Jan-35
202107	Staff	Paula	p.lopez@qmul.ac.uk	29-Jan-21
202108	Staff	Clive	c.steele@qmul.ac.uk	31-Jan-22
202109	Staff	Nenna	n.laal@qmul.ac.uk	24-Jan-25
202110	Staff	Anne	a.zhang@qmul.ac.uk	29-Jan-25
202111	Staff	Nicola	n.phillips@qmul.ac.uk	23-Jan-22
202112	Staff	Ken	ken.jones@qmul.ac.uk	21-Jan-27
202113	Staff	Sonia	s.patel@qmul.ac.uk	31-Jan-23

The MEMBER table depends on the Copies table through the attribute Member_Type.

The first two validation checks were carried out to check the format of the email addresses. The third validation check tested the unique constraint on email addresses. The next check is for the Member_Type value which is taken from the Copies table and can only be Student or Staff. The Expiry_Date was tested to ensure that it was not empty. The final check tested the unique constraint on the primary key, Member_ID.

INSERT INTO MEMBER
(Member_ID, Member_Type, Member_Name, Member_Email, Expiry_Date)
VALUES (202114, 'Staff', 'Test-1', 'test1@drgt', '31-JAN-2023')
This cannot be inserted as '.' is missing in the Member_Email.

INSERT INTO MEMBER
(Member_ID, Member_Type, Member_Name, Member_Email, Expiry_Date)
VALUES (202115, 'Staff', 'Test-2', 'test2gmail.com', '13-MAR-2023')
This cannot be inserted as '@' is missing in the Member_Email.

INSERT INTO MEMBER
(Member_ID, Member_Type, Member_Name, Member_Email, Expiry_Date)
VALUES (202116, 'Student', 'test-3', 's.patel@qmul.ac.uk', '31-DEC-2023')
This cannot be inserted as the Member_Email is repeated.

INSERT INTO MEMBER
(Member_ID, Member_Type, Member_Name, Member_Email, Expiry_Date)
VALUES (202117, 'anyone', 'Test-4', 'test4@qmul.ac.uk', '31-JAN-2023')
This cannot be inserted as the Member_Type is not in the Copies table.

INSERT INTO MEMBER
(Member_ID, Member_Type, Member_Name, Member_Email)
VALUES (202118, 'Staff', 'Test-5', 'test5@qmul.ac.uk')

This cannot be inserted as Expiry_Date cannot be NULL.

INSERT INTO MEMBER
(Member_ID, Member_Type, Member_Name, Member_Email, Expiry_Date)
VALUES (202106, 'Staff', 'Test-6', 's.patel@qmul.ac.uk', '31-JAN-2023')
This cannot be inserted as the Member_ID is repeated.

Class:

Class_No	Class_Name
101	Computing and Information Systems
102	Actuarial Science
103	Biomedical Sciences
104	Electronic Engineering
105	Law
106	Accounting and Finance
107	French and Politics
108	Finance
109	Film Studies and Drama
110	Business Management
111	Computer Science
112	Chemistry
113	History
114	Dentistry
115	Materials Science and Engineering
116	Computer Programming
117	Iridology
118	Biology
119	Forensic Science
120	Physics
121	Literature
122	Anatomy

The Class table has no dependencies.

INSERT INTO Class
(Class_Name)
VALUES ('Literature')
This cannot be inserted as the value is repeated.

INSERT INTO Class
(Class_No, Class_Name)
VALUES (5, 'Test-2')
This cannot be inserted as the Class_No is automatically created.

Res:

Resource_ ID	Resource_Title	Total_ Copies	Loan_ Period	Floor_ No	Shelf_ No	Class_ No	Resource_ Type	Author	Release_ Date	Publsh_ Drct_ Art	Edit
100001	Forensic Science	12	14	1	50	119	Book	Mike Milligan	15-Dec-14	HarperCollins	1
100002	Crime Scene to Court: The Essentials of Forensic Science	3	14	1	51	108	Book	Alice Newman	15-Dec-18	Oxford University Press	2
100003	The Law Book: Big Ideas Simply Explained	5	14	0	80	105	Book	Fatima Choudhury	15-Dec-12	Penguin	1
100004	The Human Bone Manual	7	14	0	112	105	Book	Laura Black	15-Dec-16	Cambridge Press	3
100005	Python for Data Analysis, 2e: Data Wrangling with Pandas, Numpy, and Ipython	4	2	2	25	117	Book	Cole Esher	15-Dec-14	Red House	1
100006	Python: For Beginners A Crash Course Guide To Learn Python in 1 Week	2	2	2	35	111	Book	Kat Park	15-Dec-13	HarperCollins	1
100007	The Eye: Basic Sciences in Practice	3	14	1	64	119	Book	Emmett Fox	15-Dec-09	Bloomsbury	1
100008	Science and Development of Muscle Hypertrophy	2	14	1	119	122	Book	Jenny Jones	15-Dec-04	Macmillan	4
100009	The Silent Patient	2	14	0	25	121	Book	Beau Allard	15-Dec-17	Pearson Education	1
100010	In the Midst of Life	5	14	0	141	121	Book	Carla Dela Pena	06-Jun-80	Bloomsbury	4
100011	The Mighty Muscles	4	0	0	145	105	Book	Jester Abayari	19-Jun-17	Oxford University Press	1
100012	The Forensic Laboratory	1	0	0	55	108	Book	Theresa Wolf	03-Feb-06	Red House	1
100013	The Life of Bo Jones	1	14	2	294	118	Video	-	24-Jan-02	Naomi Parker	-
100014	Cooking with Ingra	1	14	2	212	122	Video	-	05-Apr-04	Kendrick Smith	-
100015	The Human Body	3	14	2	422	122	Video	-	30-Jul-05	Jules Verner	-
100016	Skin and Bones	1	14	2	367	111	Video	-	27-Nov-01	John Adams	-
100017	The Future of AI	7	14	2	469	111	Video	-	08-Oct-01	Arthur Read	-
100018	Playing Games	1	14	2	267	122	Video	-	28-Jan-01	Ophelia Wroe	-
100019	Music Theory	1	14	2	128	122	Video	-	21-Oct-04	Nazrin Uddin	-
100020	Robotics	1	14	2	510	117	Video	-	13-Jan-06	Raksha Patel	-
100021	Paris in 5 Days	1	14	2	460	112	Video	-	06-Mar-02	Carmen Lo	-
100022	Python for Beginners	4	14	1	89	111	Video	-	09-Feb-03	Suella Burns	-
100023	WW2:True Stories	1	14	1	46	113	CD	-	18-May-17	Holly News	-
100024	Law and Order	1	14	1	63	105	CD	-	24-Feb-20	Melissa White	-
100025	War on Drugs	1	14	1	105	105	CD	-	21-Mar-18	Faizan Alim	-
100026	The Animal Kingdom	1	14	1	460	118	CD	-	14-Apr-12	Andres Luna	-
100027	A Question of Ethics	1	14	1	80	105	CD	-	04-Aug-18	Hanisha Dubasia	-
100028	American Civil Rights	1	14	1	69	105	CD	-	08-Nov-16	Montaha Behum	-
100029	Statistics	1	14	1	100	108	CD	-	14-Feb-17	Shaira Malik	-
100030	The Red Planet	1	14	1	212	120	CD	-	09-Sep-19	Lisa Jones	-
100031	Our Solar System	1	14	1	213	120	CD	-	07-May-13	Buster Baxter	-
100032	The Animal Kingdom	1	14	1	263	118	CD	-	24-Dec-15	Peggy Michaelson	-
100033	Forensic Science	1	14	1	300	119	DVD	-	16-Aug-19	Cameron Carter	-
100034	The Basics of Python	3	14	1	200	101	DVD	-	23-Mar-17	Abdul Shah	-
100035	Blood Spatter Analysis	1	14	1	62	119	DVD	-	03-Jan-18	Jessminda Patel	-
100036	Scanning Electron Microscopy	1	14	1	72	119	DVD	-	12-Oct-15	May Sung	-
100037	African Safari	1	14	1	120	101	DVD	-	06-Oct-14	Saleha Bagom	-
100038	The Goldilocks Zone	1	14	1	139	101	DVD	-	14-May-16	Nazma Hussein	-
100039	Life of Crime	1	14	1	269	101	DVD	-	21-Sep-19	Eve Valentine	-
100040	The First World War	1	14	1	216	101	DVD	-	01-Feb-18	Ronald Johnson	-
100041	The Second World War	1	14	1	217	101	DVD	-	09-Dec-19	Jeremy Coulter	-
100042	Frozen-2	1	14	1	319	121	DVD	-	17-Jul-09	Chris Buck	-
100043	Logical Coding	1	14	1	204	105	DVD	-	17-Apr-15	Poppy Sumawong	-

The Res table depends on the Class table through the attribute Class_No.

INSERT INTO Res
(Resource_Title, Loan_Period, Floor_No, Shelf_No, Class_no, Resource_Type, Publsh_Drct_Art, Release_Date)
VALUES ('Test-01', 14, 1, 216, 105, 'DVD', 'Ro John', '01-APR-18')
This cannot be inserted as Total_Copies cannot be NULL.

INSERT INTO Res
(Resource_Title, Total_Copies, Loan_Period, Floor_No, Shelf_No, Class_no, Resource_Type, Publsh_Drct_Art, Release_Date)
VALUES ('Test-02', 5, 16, 1, 217, 118, 'CD', 'Jerey Kapoor', '09-Mar-16')
This cannot be inserted as the value of Loan_Period can only be 0, 2 or 14, not 16.

INSERT INTO Res
(Resource_Title, Total_Copies, Loan_Period, Floor_No, Shelf_No, Class_no, Resource_Type, Author, Publsh_Drct_Art, Release_Date, edit)
VALUES ('test-03', 7, 2, 1, 50, 102, 'Book','Pawan Sharma', 'abc co', '29-May-10',1)
This cannot be inserted as the combination of Floor_No and Shelf_No is repeated.

INSERT INTO Res
(Resource_Title, Total_Copies, Loan_Period, Floor_No, Shelf_No, Class_no, Resource_Type, Author, Publsh_Drct_Art, Release_Date, Edit)
VALUES ('Test-04', 5, 14, 0, 81, 105, 'ebook', 'Fatima Choudhury', 'Penguin', '15-JUN-12', 1)
This cannot be inserted as the value of Resource_Type cannot be ebook.

INSERT INTO Res
(Resource_Title, Total_Copies, Loan_Period, Floor_No, Shelf_No, Class_no, Resource_Type, Release_Date)
VALUES ('Test-05', 1, 14, 0, 267, 122, 'Video', '28-JAN-17')
This cannot be inserted as Publsh_Drct_Art cannot be NULL.

Borrow:

Member_ID	Resource_ID	Due_Date	Loan_ID
202005	100036	02-Sep-20	1000001
202002	100001	04-Sep-20	1000002
202003	100002	11-Sep-20	1000003
202004	100001	07-Sep-20	1000004
202005	100002	10-Sep-20	1000005
202006	100003	13-Sep-20	1000006
202007	100004	08-Sep-20	1000007
202008	100005	03-Sep-20	1000008
202009	100006	06-Oct-20	1000009
202010	100007	11-Oct-20	1000010
202012	100008	07-Oct-20	1000011
202013	100010	13-Oct-20	1000012
202014	100009	09-Nov-20	1000013
202015	100008	06-Nov-20	1000014

202101	100007	14-Nov-20	1000015
202102	100006	09-Nov-20	1000016
202103	100012	12-Nov-20	1000017
202104	100017	08-Dec-20	1000018
202106	100023	05-Dec-20	1000019
202106	100035	05-Dec-20	1000020
202111	100013	05-Dec-20	1000021
202011	100015	06-Dec-20	1000022
202012	100021	07-Dec-20	1000023
202106	100003	09-Dec-20	1000024
202007	100019	10-Dec-20	1000025
202110	100032	01-Dec-20	1000026
202109	100040	03-Dec-20	1000027
202012	100033	06-Dec-20	1000028
202013	100028	08-Dec-20	1000029
202001	100025	09-Dec-20	1000030
202004	100026	11-Dec-20	1000031
202104	100017	11-Dec-20	1000032
202105	100019	12-Dec-20	1000033
202011	100039	12-Dec-20	1000034
202012	100037	12-Dec-20	1000035
202015	100018	12-Dec-20	1000036
202113	100023	16-Dec-20	1000037
202006	100003	17-Dec-20	1000038
202009	100007	20-Dec-20	1000039
202109	100032	20-Dec-20	1000040

The Borrow table depends on the Res and MEMBER tables through the attributes Resource_ID and Member_ID, respectively.

INSERT INTO Borrow
(Member_ID, Resource_ID, Due_Date)
VALUES (202200, 100025, '09-DEC-20')
This cannot be inserted as the Member_ID does not exist.

INSERT INTO Borrow
(Member_ID, Resource_ID)
VALUES (202200, 100025)
This cannot be inserted as the Due_Date cannot be NULL.

INSERT INTO Borrow
(Member_ID, Resource_ID, Due_Date)
VALUES (202202, 10002, '09-SEP-20')
This cannot be inserted as the Resource_ID does not exist.

Return:

Loan_ID	Return_Date
1000002	02-Sep-20
1000004	07-Sep-20
1000007	08-Sep-20
1000005	10-Sep-20
1000006	20-Sep-20
1000009	06-Oct-20
1000011	06-Oct-20
1000014	08-Nov-20
1000013	09-Nov-20
1000017	12-Nov-20
1000015	14-Nov-20
1000012	30-Nov-20
1000025	03-Dec-20
1000023	04-Dec-20
1000019	05-Dec-20
1000021	05-Dec-20
1000020	06-Dec-20
1000022	06-Dec-20
1000040	06-Dec-20
1000018	08-Dec-20
1000024	09-Dec-20
1000026	09-Dec-20
1000027	30-Nov-20
1000028	06-Dec-20
1000029	05-Dec-20
1000030	11-Dec-20
1000031	08-Dec-20
1000032	10-Dec-20
1000033	13-Dec-20
1000034	12-Dec-20
1000035	11-Dec-20
1000036	12-Dec-20
1000037	12-Dec-20

The Return table depends on the Borrow table through the common attribute Loan_ID.

INSERT INTO RETURN
(Loan_ID, Return_Date)
VALUES (1000043, '09-MAR-20')
This cannot be inserted as the Loan_ID does not exist.

INSERT INTO RETURN
(Loan_ID)

VALUES (1000029)
This cannot be inserted as the Return_Date cannot be NULL.

INSERT INTO RETURN
(Loan_ID, Return_Date)
VALUES (1000013, '22-JUN-20')
This cannot be inserted as the Loan_ID is repeated.

Fine_Payment:

Loan_ID	Date_Paid	Amount_Paid
1000006	20-Sep-20	7
1000012	02-Dec-20	48
1000014	08-Nov-20	2
1000020	08-Dec-20	1

The Fine_Payment table depends on the Borrow table through the common attribute Loan_ID.

INSERT INTO Fine_Payment
(Loan_ID, Date_Paid, Amount_Paid)
VALUES (1000059, '20-DEC-20',7)
This cannot be inserted as the Loan_ID does not exist.

INSERT INTO Fine_Payment
(Loan_ID, Amount_Paid)
VALUES (1000006, 7)
This cannot be inserted as Date_Paid cannot be NULL.

INSERT INTO Fine_Payment
(Loan_ID, Date_Paid)
VALUES (1000006, '2-MAR-20')
This cannot be inserted as Amount_Paid cannot be NULL.

Enrol:

Member_ID	Class_No	Enrol_Date
202001	118	24-Sep-19
202002	115	19-Sep-20
202003	114	24-Sep-19
202004	119	24-Sep-19
202005	120	24-Sep-17
202006	113	24-Sep-19
202007	112	24-Sep-20
202008	111	24-Sep-20
202009	110	24-Sep-19
202010	109	24-Sep-19

The Enrol table depends on the MEMBER and Class tables through the attributes Member_ID and Class_No, respectively.

INSERT INTO Enrol
(Member_ID, Class_No, Enrol_Date)
VALUES (202011, 186, '14-JUN-20')
This cannot be inserted as Class_No does not exist.

INSERT INTO Enrol
(Member_ID, Class_No, Enrol_Date)
VALUES (232012, 109, '24-JAN-20')
This cannot be inserted as Member_ID does not exist.

INSERT INTO Enrol
(Member_ID, Class_No)
VALUES (202010, 109)
This cannot be inserted as Enrol_Date cannot be NULL.

Fine - Derived Table

```
CREATE Table Fine AS
/* Getting Details of LoanId, MemberId, resourceID and Total Fine in case of returns */
SELECT Borrow.Loan_ID as LoanID, Member_ID as MID, Resource_ID as RID, (RETURN_DATE
- Due_Date)*1  AS Fine, Return_Date as OnDate
FROM Return, Borrow
WHERE Borrow.Loan_ID = Return.Loan_ID AND (Return_date - Due_Date)*1 > 0
UNION
/* Uniting the above with Total fine if it is not returned and passed the due date by
comparing against system Date*/
SELECT Borrow.Loan_ID AS LoanID, Member_ID as MID, Resource_ID as RID,
(to_date(sysdate) - Due_DATE) AS Fine, to_date(sysdate) AS OnDate from Borrow
where not Borrow.Loan_ID = any(select Loan_id from return) AND
to_date(sysdate)>due_date;
/* Updating the fine and calculating the fine in case of paid fine */
UPDATE fine SET fine = Fine - NVL((select amount_paid from fine_payment
where fine.loanid = fine_payment.Loan_ID), 0);
DELETE Fine WHERE Fine = 0;
select * from fine;
```

LOANID	MID	RID	FINE	ONDATE
1000001	202005	100036	102	12-Dec-20
1000003	202003	100002	93	12-Dec-20
1000008	202008	100005	101	12-Dec-20
1000010	202010	100007	63	12-Dec-20
1000016	202102	100006	34	12-Dec-20
1000026	202110	100032	8	09-Dec-20
1000030	202001	100025	2	11-Dec-20
1000033	202105	100019	1	13-Dec-20

A derived table Fine was created to assist with the views, queries and triggers. The table included the columns Loan_ID, as LOANID, Member_ID as MID, Resource_ID as RID, FINE and ONDATE.

FINE is a derived attribute who's values are calculated by multiplying the number of days overdue with the fine per day, which is $1. The number of days overdue is calculated by subtracting Return_Date (Return table) from Due_Date (Borrow table) in case of the resource being returned. If the resource is not returned, the number of days overdue is calculated by subtracting the system date (sysdate) from Due_Date.

The values of fine in the Fine table is further updated by subtracting Amount_Paid from Fine_Payment. If the Fine value is 0, then the row is deleted.

Views

List of suspended members:

```
CREATE OR REPLACE VIEW SUSPEND AS
SELECT a.MID, b.M_Name, Email, a.Overdues, TotalFine FROM
        (SELECT MID, Count(LID1) AS Overdues FROM
           (SELECT MID, LoanID AS LID1 FROM
            fine
           WHERE NOT(LoanID = any(SELECT LOAN_ID FROM RETURN)))
         GROUP BY MID) a, /*Getting a list of all members and the count of unreturned
resources*/
        (SELECT Member_ID, Member_Name AS M_Name, Member_Email AS Email
         FROM Member) b, /*Get member details from Member Table*/
        (SELECT MID, Sum(Fine) AS TotalFine FROM Fine Group By MID HAVING Sum(Fine)
>= 10) c  /*To check that if the sum of fines is greater than 10*/
WHERE a.MID = b.Member_ID AND a.MID = c.MID;
```

This view can be used by both library staff and members. Library staff will have ready access to the list of currently suspended members and will not have to go through the process of filtering out the desired data from the raw data. This will help in data security as the view is more like a virtual table. The view can also be used by members to get their details or when checking whether or not they have been suspended.

MID	M_NAME	EMAIL	OVERDUES	TOTALFINE
202003	Purvi	purvimandodadoda@hotmail.co.uk	1	93
202005	Hana	hanarahman@hotmail.com	1	102
202008	Cef	cefalrahman@hotmail.com	1	101
202010	Holly	hollylou@outlook.co.uk	1	63
202102	Mahesha	m.soong@qmul.ac.uk	1	34

Available_Copies and Popularity:

```
Create OR REPLACE View Full_Res As
  /* Adding details of popularity (Number of borrows) and Availability (Total copies -
borrows + returns) in the library system */
Select x.Resource_ID, Resource_Title, Loan_Period, Floor_No, Shelf_No, Class_no,
Resource_Type, Author, Publsh_Drct_Art, Release_Date, Edit,
      Total_Copies, (Total_Copies - NVL(CurrentOut,0)) as Available_Copies, NVL(Popularity,
0) AS POPULARITY From
 /* Selecting all resources and joining them to get count of borrowed and returned
resources */
   (Select * From Res) x LEFT OUTER JOIN
     /* Joining the Borrowed Resources and Returned resources queries against
Resource_Ids, It will have all borrowed resources*/
```

```
    (Select a.Resource_ID, (Borrows - NVL(Returns,0)) AS CurrentOut, Borrows AS Popularity
From
        /* Counting the Number of borrowed Against Resource IDs*/
       (Select Resource_ID, Count(Loan_ID) AS Borrows FROM Borrow Group By
Resource_ID) a LEFT OUTER JOIN
      /* Counting the number of returned Resource_Ids */
        (SELECT Resource_ID, Count(Loan_ID) AS Returns FROM
           /* Creating a query for of all returned Loan Ids and related Resource_IDs */
           (SELECT unique Borrow.Loan_ID, Resource_ID FROM BORROW, Return
         WHERE Return.Loan_ID = Borrow.Loan_ID)
        GROUP BY Resource_ID) b
    ON a.Resource_ID = b.Resource_ID) y
ON x.Resource_ID = y.Resource_ID
Order BY resource_Id;
```

This view was created to show Available_Copies and Popularity of each resource. Available_Copies refers to the number of copies which are available for each resource. Popularity is defined by the number of times a particular resource is borrowed. This will help members check the status of desired resources and how popular they are with other members. The library staff can use this information to order more of the resources which are popular and possibly lower the loan period so that more members can get access to them. This could also be used to update the recommended resources for classes. For resources that been borrowed the least or not at all, library staff can have them removed. The library staff can also use the Available_Copies to audit the number of resources in the library.

Resource_ ID	Resource_Title	Loan_ Period	Floor_ No	Shelf_ No	Class_ No	Resource_ Type	Author	Publsh_Drct_Art	Release_Date	Edit	Total_ Copies	Available_ Copies	Popularity
100001	Forensic Science	14	1	50	119	Book	Mike Milligan	HarperCollins	15-Dec-14	1	12	12	2
100002	Crime Scene to Court: The Essentials of Forensic Science	14	1	51	108	Book	Alice Newman	Oxford University Press	15-Dec-18	2	3	2	2
100003	The Law Book: Big Ideas Simply Explained	14	0	80	105	Book	Fatima Choudhury	Penguin	15-Dec-12	1	5	4	3
100004	The Human Bone Manual	14	0	112	105	Book	Laura Black	Cambridge Press	15-Dec-16	3	7	7	1
100005	Python for Data Analysis, 2e: Data Wrangling with Pandas, Numpy, and Ipython	2	2	25	117	Book	Cole Esher	Red House	15-Dec-14	1	4	3	1
100006	Python: For Beginners A Crash Course Guide To Learn Python in 1 Week	2	2	35	111	Book	Kat Park	HarperCollins	15-Dec-13	1	2	1	2
100007	The Eye: Basic Sciences in Practice	14	1	64	119	Book	Emmett Fox	Bloomsbury	15-Dec-09	1	3	1	3
100008	Science and Development of Muscle Hypertrophy	14	1	119	122	Book	Jenny Jones	Macmillan	15-Dec-04	4	2	2	2
100009	The Silent Patient	14	0	25	121	Book	Beau Allard	Pearson Education	15-Dec-17	1	2	2	1
100010	In the Midst of Life	14	0	141	121	Book	Carla Dela Pena	Bloomsbury	06-Jun-80	4	5	5	1
100011	The Mighty Muscles	0	0	145	105	Book	Jester Abayari	Oxford University Press	19-Jun-17	1	4	4	0
100012	The Forensic Laboratory	0	0	55	108	Book	Theresa Wolf	Red House	03-Feb-06	1	1	1	1
100013	The Life of Bo Jones	14	2	294	118	Video	-	Naomi Parker	24-Jan-02	-	1	1	1
100014	Cooking with Ingra	14	2	212	122	Video	-	Kendrick Smith	05-Apr-04	-	1	1	0
100015	The Human Body	14	2	422	122	Video	-	Jules Verner	30-Jul-05	-	3	3	1
100016	Skin and Bones	14	2	367	111	Video	-	John Adams	27-Nov-01	-	1	1	0
100017	The Future of AI	14	2	469	111	Video	-	Arthur Read	08-Oct-01	-	7	7	2
100018	Playing Games	14	2	267	122	Video	-	Ophelia Wroe	28-Jan-01	-	1	1	1
100019	Music Theory	14	2	128	122	Video	-	Nazrin Uddin	21-Oct-04	-	1	1	2
100020	Robotics	14	2	510	117	Video	-	Raksha Patel	13-Jan-06	-	1	1	0
100021	Paris in 5 Days	14	2	460	112	Video	-	Carmen Lo	06-Mar-02	-	1	1	1
100022	Python for Beginners	14	1	89	111	Video	-	Suella Burns	09-Feb-03	-	4	4	0
100023	WW2:True Stories	14	1	46	113	CD	-	Holly News	18-May-17	-	1	1	2
100024	Law and Order	14	1	63	105	CD	-	Melissa White	24-Feb-20	-	1	1	0
100025	War on Drugs	14	1	105	105	CD	-	Faizan Alim	21-Mar-18	-	1	1	1
100026	The Animal Kingdom	14	1	460	118	CD	-	Andres Luna	14-Apr-12	-	1	1	1
100027	A Question of Ethics	14	1	80	105	CD	-	Hanisha Dubasia	04-Aug-18	-	1	1	0
100028	American Civil Rights	14	1	69	105	CD	-	Montaha Behum	08-Nov-16	-	1	1	1
100029	Statistics	14	1	100	108	CD	-	Shaira Malik	14-Feb-17	-	1	1	0
100030	The Red Planet	14	1	212	120	CD	-	Lisa Jones	09-Sep-19	-	1	1	0
100031	Our Solar System	14	1	213	120	CD	-	Buster Baxter	07-May-13	-	1	1	0
100032	The Animal Kingdom	14	1	263	118	CD	-	Peggy Michaelson	24-Dec-15	-	1	1	2
100033	Forensic Science	14	1	300	119	DVD	-	Cameron Carter	16-Aug-19	-	1	1	1
100034	The Basics of Python	14	1	200	101	DVD	-	Abdul Shah	23-Mar-17	-	3	3	0
100035	Blood Spatter Analysis	14	1	62	119	DVD	-	Jessminda Patel	03-Jan-18	-	1	1	1
100036	Scanning Electron Microscopy	14	1	72	119	DVD	-	May Sung	12-Oct-15	-	1	0	1
100037	African Safari	14	1	120	101	DVD	-	Saleha Bagom	06-Oct-14	-	1	1	1
100038	The Goldilocks Zone	14	1	139	101	DVD	-	Nazma Hussein	14-May-16	-	1	1	0
100039	Life of Crime	14	1	269	101	DVD	-	Eve Valentine	21-Sep-19	-	1	1	1
100040	The First World War	14	1	216	101	DVD	-	Ronald Johnson	01-Feb-18	-	1	1	1
100041	The Second World War	14	1	217	101	DVD	-	Jeremy Coulter	09-Dec-19	-	1	1	0
100042	Frozen-2	14	1	319	121	DVD	-	Chris Buck	17-Jul-09	-	1	1	0
100043	Logical Coding	14	1	204	105	DVD	-	Poppy Sumawong	17-Apr-15	-	1	1	0

List of borrows with status:

```
CREATE OR REPLACE VIEW full_borrow AS
Select a.Loan_ID as Loan_ID, Member_ID As Member_ID, Due_Date AS Due_On,
Resource_ID As RID, NVL(Status,'Current') As Status From
   (Select Loan_ID, Member_ID, Due_Date, Resource_ID From Borrow) a LEFT OUTER JOIN
   (Select Loan_ID, 'Previous' AS Status From Borrow Where
         Borrow.Loan_Id = any(select Loan_id from Return)) b
   ON
   a.loan_ID = b.loan_ID;
```

This view was created to show the status of a loan, whether it is a current or previous loan. This will assist both members and library staff to check the details of all loans. The view will also help with queries regarding borrows.

Loan_ID	Member_ID	Due_On	RID	STATUS
1000001	202005	02-Sep-20	100036	Current
1000002	202002	04-Sep-20	100001	Previous
1000003	202003	11-Sep-20	100002	Current
1000004	202004	07-Sep-20	100001	Previous
1000005	202005	10-Sep-20	100002	Previous
1000006	202006	13-Sep-20	100003	Previous
1000007	202007	08-Sep-20	100004	Previous
1000008	202008	03-Sep-20	100005	Current
1000009	202009	06-Oct-20	100006	Previous
1000010	202010	11-Oct-20	100007	Current
1000011	202012	07-Oct-20	100008	Previous
1000012	202013	13-Oct-20	100010	Previous
1000013	202014	09-Nov-20	100009	Previous
1000014	202015	06-Nov-20	100008	Previous
1000015	202101	14-Nov-20	100007	Previous
1000016	202102	09-Nov-20	100006	Current
1000017	202103	12-Nov-20	100012	Previous
1000018	202104	08-Dec-20	100017	Previous
1000019	202106	05-Dec-20	100023	Previous
1000020	202106	05-Dec-20	100035	Previous

Queries

Current members based on Member_Type:

This query can be used to see details of either Student or Staff members. This information can be used when sending notices to either all Student members or all Staff members.

```
SELECT Member_ID, Member_Name, Member_Email, Expiry_Date
FROM MEMBER
WHERE Member_Type = 'Student';
```

Member_ID	Member_Name	Member_Email	Expiry_Date
202001	Shajeda	shajedasyed@yahoo.com	23-Jan-23
202002	Kayley	kanett@hotmail.co.uk	28-Jan-24
202003	Purvi	purvimandodadoda@hotmail.co.uk	21-Jan-23
202004	Steven	stevencorpuz@gmail.co.uk	14-Jan-23
202005	Hana	hanarahman@hotmail.com	29-Jan-21
202006	Rohit	rohitgarg12@hotmail.com	14-Jan-23
202007	Tanya	tanyasyed_@yahoo.com	20-Jan-24
202008	Cef	cefalrahman@hotmail.com	22-Jan-24
202009	Daniel	debenham_dan@hotmail.co.uk	25-Jan-23
202010	Holly	hollylou@outlook.co.uk	30-Jan-23
202011	Rafayet	rafayettarafder@live.co.uk	21-Jan-22
202012	Ope	oppeeeee@outlook.co.uk	21-Jan-22
202013	Jack	jack_1995@hotmail.co.uk	25-Jan-23
202014	Teddy	teddy.moonshine@live.co.uk	29-Jan-22
202015	Maria	maria.678@gmail.com	21-Jan-21

Resources available for a particular class:

This query list all resources that are available for a particular class. Members can use this query to look up resources that they can use for each of their classes.

```
SELECT Resource_ID, Resource_Title, Resource_Type, Class_Name
FROM Res, Class
Where Res.Class_No = Class.Class_No AND Class.Class_Name = 'Physics';
```

Resource_ID	Resource_Title	Resource_Type	Class_Name
100030	The Red Planet	CD	Physics
100031	Our Solar System	CD	Physics

Most popular Author:

This query can be used to find out which 5 authors are the most popular amongst members. Library staff can use this information to buy more books written by those authors.

```
SELECT Res.Author, COUNT (Res.Author) as Times_Borrowed
FROM Borrow
INNER JOIN Res ON Res.Resource_ID = Borrow.Resource_ID
GROUP BY Res.Author
ORDER BY Times_Borrowed DESC
FETCH NEXT 5 ROWS ONLY;
```

Author	Times_Borrowed
Fatima Choudhury	3
Emmett Fox	3
Kat Park	2
Mike Milligan	2
Alice Newman	2

Resource_Title based resource search:

This query represents a method of finding a resource based on its title. Members can use this to find specific resources.

```
Select * From Full_Res
Where Resource_Title LIKE '%Forensic Science';
```

Resource_ ID	Resource_Title	Loan_ Period	Floor_ No	Shelf_ No	Class_ No	Resource_ Type	Author	Publsh_ Drct_Art	Release_ Date	Edit	Total_ Copies	Available_ Copies	Popularity
100001	Forensic Science	14	1	50	119	Book	Mike Milligan	HarperCollins	15-Dec-14	1	12	12	2
100002	Crime Scene to Court: The Essentials of Forensic Science	14	1	51	108	Book	Alice Newman	Oxford University Press	15-Dec-18	2	3	2	2
100033	Forensic Science	14	1	300	119	DVD	-	Cameron Carter	16-Aug-19	-	1	1	1

Author based resource search:

This query represents a method of finding a resource based on its author. Members can use this to find specific resources.

```
Select * From Full_Res
Where Author LIKE '%Mike%';
```

Resource_ ID	Resource_Title	Loan_ Period	Floor_ No	Shelf_ No	Class_ No	Resource_ Type	Author	Publsh_ Drct_Art	Release_ Date	Edit	Total_ Copies	Available_ Copies	Popularity
100001	Forensic Science	14	1	50	119	Book	Mike Milligan	HarperCollins	15-Dec-14	1	12	12	2

Number of resources based on Resource_Type:

This query lists the number of resources based on Resource_Type. This can library staff to decide which resource types are more than required to procure future resources.

```
SELECT count(Resource_ID), Resource_Type
FROM Res
group by (resource_type);
```

COUNT(RESOURCE_ID)	RESOURCE_TYPE
11	DVD
12	Book
10	Video
10	CD

Members that currently have fines and the fine amount:

This query lists the fines of members. It will help library staff to see the members who have been defaulting.

```
SELECT Borrow.Member_ID, Member_Name, Member_Email, Loan_ID, Fine
FROM
Member, Borrow, Fine
WHERE
Borrow.Member_ID = Member.Member_ID AND Fine.LOanID = Borrow.Loan_ID
ORDER BY Fine DESC;
```

Member_ID	Member_Name	Member_Email	Loan_ID	Fine
202005	Hana	hanarahman@hotmail.com	1000001	102
202008	Cef	cefalrahman@hotmail.com	1000008	101
202003	Purvi	purvimandodadoda@hotmail.co.uk	1000003	93
202010	Holly	hollylou@outlook.co.uk	1000010	63
202102	Mahesha	m.soong@qmul.ac.uk	1000016	34
202110	Anne	a.zhang@qmul.ac.uk	1000026	8
202001	Shajeda	shajedasyed@yahoo.com	1000030	2
202105	Mark	m.amaral@qmul.ac.uk	1000033	1

Resources that have not been returned:

This query lists all the resources which have not yet been returned. Library staff can use this information to follow up with members.

```
SELECT UNIQUE Full_borrow.RID, Resource_Title
FROM Full_borrow, Res
WHERE Full_borrow.RID = Res.Resource_ID AND Full_borrow.status = 'Current';
```

RID	Resource_Title

100003	The Law Book: Big Ideas Simply Explained
100006	Python: For Beginners A Crash Course Guide To Learn Python in 1 Week
100007	The Eye: Basic Sciences in Practice
100036	Scanning Electron Microscopy
100002	Crime Scene to Court: The Essentials of Forensic Science
100005	Python for Data Analysis, 2e: Data Wrangling with Pandas, Numpy, and Ipython

Number of resources based on Class_Name:

This query lists the number of resources the library has for each class. This information can be used to procure more resources for classes will lesser resources.

```
SELECT Class_Name, NVL(NoOfResources,0) AS NoOfResources From
 (SELECT Class_No, Class_Name From Class) a LEFT OUTER JOIN
 (Select count(Resource_ID) AS NoOfResources, Class_No From Res Group BY Class_No) b
ON a.Class_No = b.Class_No
Order By Class_Name;
```

Class_Name	NoOfResources
Accounting and Finance	0
Actuarial Science	0
Anatomy	5
Biology	3
Biomedical Sciences	0
Business Management	0
Chemistry	1
Computer Programming	0
Computer Science	4
Computing and Information Systems	6
Dentistry	0
Electronic Engineering	0
Film Studies and Drama	0
Finance	3
Forensic Science	5
French and Politics	0
History	1
Iridology	2
Law	8
Literature	3
Materials Science and Engineering	0
Physics	2

Total Fine collection within a time period:

This query gives the sum of the amount collected in fines by the library within a specific time period. This will help in the auditing of finance flow.

```
Select SUM (Amount_Paid)
FROM Fine_Payment
WHERE Date_Paid
BETWEEN '01-DEC-20' AND '12-DEC-20';
```

SUM(AMOUNT_PAID)
49

Member Status and Fine based on Member_Email:

This query lists member status, whether active or suspended. It also lists the fine amount that is due. This query will help members to view their details.

```
SELECT Member.Member_ID, Member.Member_Name, Member.Expiry_Date, c.STATUS,
c.fine FROM
   Member,
   (SELECT a.Member_ID, NVL(b.Member_Status, 'ACTIVE') AS STATUS, fine FROM
      (SELECT Member_ID FROM MEMBER) a LEFT OUTER JOIN
      (SELECT MID as Member_ID, TotalFine AS Fine, 'SUSPEND' AS Member_Status FROM
Suspend) b
   ON a.Member_ID = b.Member_ID) c
WHERE Member.Member_ID = c.Member_ID and Member_Email =
'tanyasyed_@yahoo.com';
```

Member_ID	Member_Name	Expiry_Date	Status	Fine
202007	Tanya	20-Jan-24	ACTIVE	-

Past and present loans of a member based on Member_Email:

This query lists all the loans of a member. If the loan has been returned, it also gives the date of return and if the loan is overdue, it shows the fine amount. This will help the member in querying and viewing present and past loans.

```
SELECT Loan_ID, Resource_ID, Due_Date, Fine, Return_Date FROM
(SELECT f.Loan_ID, Member_ID, Resource_ID, Due_Date, Fine, Return_Date FROM
(SELECT d.Loan_ID, Resource_ID, Member_ID, Due_Date, Return_Date FROM
   (SELECT Loan_ID, Resource_ID, Member_ID, Due_Date from Borrow) d LEFT OUTER JOIN
   (SELECT Loan_ID, Return_Date From Return) e
ON d.Loan_ID = e.Loan_ID) f LEFT OUTER JOIN
(SELECT a.Loan_ID, Fine  From
    (SELECT Loan_ID From Borrow) a LEFT OUTER JOIN
    (SELECT LoanID, Fine From Fine) b
 ON a.Loan_ID = b.LoanID) c
```

```
ON f.Loan_ID = c.Loan_ID) k, Member
WHERE Member.Member_ID = k.Member_ID and Member.Member_Email =
'rohitgarg12@hotmail.com';
```

Loan_ID	Resource_ID	Due_Date	Fine	Return_Date
1000006	100003	13-Sep-20	-	20-Sep-20
1000038	100003	17-Dec-20	-	-

Triggers

1. Borrow:

```
create or replace trigger borrow_valid
before insert on borrow
for each row
declare
pd res.loan_period%type;
NoCurrent number;
mtype member.member_type%type;
maxloans copies.copies_allowed%type;
exp_date member.expiry_date%type;
suspen number;
fine number;
maxtime res.loan_period%type;
Avail Full_Res.Available_Copies%type;
begin
   select count(*) into NoCurrent from borrow where (not(loan_id = any(select loan_id from
return))) and member_id = :new.member_id;
 select loan_period into pd from res where resource_id = :new.resource_id;
 select member_type into mtype from member where member_id = :new.member_id;
 select expiry_date into exp_date from member where member_id = :new.member_id;
 select copies_allowed into maxloans from copies where copies.member_type = mtype;
 select count(*) into suspen from suspend where :new.member_id = any(select mid from
suspend);
 Select Loan_Period into maxtime from Res where :new.Resource_id = Res.Resource_id;
 Select Available_Copies Into Avail from Full_Res where :new.Resource_ID =
Full_Res.Resource_ID;
 :new.due_date := to_date (sysdate) + pd;

 if NoCurrent >= maxloans THEN
 raise_application_error (-20134,'Already have maximum current loans');
 else
  if exp_date < to_date (sysdate) + pd THEN
  raise_application_error (-20135,'Due date exceed your expiry date');
        else
                if suspen != 0 then
          raise_application_error (-20136, 'Account Suspended - Return resource or pay full
fine');
                else
                        if maxtime = 0 Then
                        raise_application_error (-20137, 'The resource cannot be borrowed');
        else
             if Avail = 0 Then
             raise_application_error (-20138, 'The resource is not available');
             end if;
```

```
                    end if;
                end if;
    end if;

  end if;
end;
/
```

The trigger does the following:

- Automatically creates Due_Date values in the Borrow entry.
- Restricts the number of resources a member can loan at a time based on whether they are a Student or Staff
- Checks if membership status will expire before the Due_Date, stops the borrow if membership will expire before
- Stops the borrow if member is suspended
- Stops the borrow if the resource can only be used inside the library (Loan_Period = 0)
- Stops the borrow if no more copies of the resource are available (Available_Copies = 0)

Automatically creates Due_Date values in the borrow entry:

```
INSERT INTO Borrow (Member_ID, Resource_ID)
VALUES (202004, 100003);

SELECT * FROM Borrow
WHERE Member_ID = 202004;
```

Member_ID	Resource_ID	Due_Date	Loan_ID
202004	100003	26-Dec-20	1000042
202004	100001	07-Sep-20	1000004
202004	100026	11-Dec-20	1000031

In the above table new Borrow values (202004, 100003) have automatically created the Due_Date value (today's date (12/DEC/20) + Loan_Period (14)).

Restricts the number of resources a member can loan at a time based on whether they are Student or Staff:

```
insert into borrow (member_id, resource_id) values (202012, 100008);
insert into borrow (member_id, resource_id) values (202012, 100009);
insert into borrow (member_id, resource_id) values (202012, 100010);
insert into borrow (member_id, resource_id) values (202012, 100013);
insert into borrow (member_id, resource_id) values (202012, 100015);
insert into borrow (member_id, resource_id) values (202012, 100019);
```

1 row(s) inserted.

1 row(s) inserted.
1 row(s) inserted.
1 row(s) inserted.
1 row(s) inserted.

ORA-20134: Already have maximum current loans ORA-06512: at "SQL_GUFJPPTKOBZXSLVMGYMMKTVIW.BORROW_VALID", line 23 ORA-06512: at "SYS.DBMS_SQL", line 1721

As the member with member_id = 202012 is a student and can loan maximum of 5 resources at a time, the trigger throws an error if there is another borrow request after 5 borrows.

Checks if membership status will expire before the Due_Date, stops the borrow if membership will expire before:

```
UPDATE MEMBER SET Expiry_Date = '14-dec-20'
WHERE Member_ID = 202010;

INSERT INTO Borrow (Member_ID, Resource_ID)
VALUES (202010, 100009);

SELECT * FROM Borrow
WHERE MEMBER_ID = 202010
```

ORA-20135: Due date exceed your expiry date ORA-06512: at "SQL_VUSAXMVBRKCWAIBZGOVJLAQJF.BORROW_VALID", line 26 ORA-06512: at "SYS.DBMS_SQL", line 1721

Member_ID	Resource_ID	Due_Date	Loan_ID
202010	100007	11-Oct-20	1000010

To check, the Expiry_Date of one of the member's was updated to 14th December 2020. Insertion of values (202010, 100009) was attempted in the Borrow table and as seen above, it failed because the Due_Date exceeds the Expiry_Date.

Stops the borrow if member is suspended:

```
select * from SUSPEND;

INSERT INTO Borrow (Member_ID, Resource_ID)
VALUES (202003, 100007);
```

MID	M_NAME	EMAIL	OVERDUES	TOTALFINE
202003	Purvi	purvimandodadoda@hotmail.co.uk	1	93
202005	Hana	hanarahman@hotmail.com	1	102
202008	Cef	cefalrahman@hotmail.com	1	101
202010	Holly	hollylou@outlook.co.uk	1	63
202102	Mahesha	m.soong@qmul.ac.uk	1	34

Download CSV
5 rows selected.

```
ORA-20136: Account Suspended - Return resource or pay full fine ORA-06512: at "SQL_VUSAXMVBRKCWAIBZGOVJLAQJF.BORROW_VALID", line 29
ORA-06512: at "SYS.DBMS_SQL", line 1721
```

Stops the borrow if the resource can only be used inside the library (Loan_Period = 0):
INSERT INTO Borrow (Member_ID, Resource_ID)
VALUES (202002, 100011);

Select * from Res;

Resource_ID	Resource_Title	Total_Copies	Loan_Period
100001	Forensic Science	12	14
100002	Crime Scene to Court: The Essentials of Forensic Science	3	14
100003	The Law Book: Big Ideas Simply Explained	5	14
100004	The Human Bone Manual	7	14
100005	Python for Data Analysis, 2e: Data Wrangling with Pandas, Numpy, and Ipython	4	2
100006	Python: For Beginners A Crash Course Guide To Learn Python in 1 Week	2	2
100007	The Eye: Basic Sciences in Practice	3	14
100008	Science and Development of Muscle Hypertrophy	2	14
100009	The Silent Patient	2	14
100010	In the Midst of Life	5	14
100011	**The Mighty Muscles**	**4**	**0**
100012	The Forensic Laboratory	1	0
100013	The Life of Bo Jones	1	14
100014	Cooking with Ingra	1	14

ORA-20137: The resource cannot be borrowed ORA-06512: at "SQL_VUSAXMVBRKCWAIBZGOVJLAQJF.BORROW_VALID", line 32 ORA-06512: at "SYS.DBMS_SQL", line 1721

Stops the borrow if no more copies of the resource are available (Available_Copies = 0):
SELECT Resource_ID, Available_Copies FROM FULL_RES where Available_Copies = 0;

INSERT INTO Borrow (Member_ID, Resource_ID)
VALUES (202012, 100007);

Resource_ID	Available_Copies
100007	0
100014	0
100019	0
100036	0

ORA-20138: The resource is not available ORA-06512: at "SQL_VUSAXMVBRKCWAIBZGOVJLAQJF.BORROW_VALID", line 35 ORA-06512: at "SYS.DBMS_SQL", line 1721

2. Return Table:

```
create or replace trigger systemdate_return
before insert on Return
for each row
begin
   IF :new.Return_Date IS NULL THEN
      :new.Return_Date := sysdate;
   END IF;
end;
/
```

The trigger creates Return_Date in the Return table.

For the demonstration:
insert into return (Loan_ID) Values (1000051);
select * from return;

OUTPUT:

Loan_ID	Return_Date
1000051	**12-Dec-20**
1000046	12-Dec-20
1000002	02-Sep-20
1000004	07-Sep-20
1000007	08-Sep-20
1000005	10-Sep-20
1000006	20-Sep-20

1000009	06-Oct-20
1000011	06-Oct-20
1000014	08-Nov-20
1000013	09-Nov-20
1000017	12-Nov-20
1000015	14-Nov-20

3. Before insert in Fine_Payment Table:

```
create or replace trigger fine_pay_check
before insert on fine_payment
for each row
declare
times number;
datedue Borrow.Due_Date%type;
datereturn Return.Return_date%type;
begin
   :new.Date_Paid := to_date(sysdate);
   select count(*) into times from return where :new.Loan_id = return.loan_id;
   select due_Date Into datedue from borrow where :new.Loan_id = borrow.loan_id;
   if times = 0 then
   raise_application_error(-20321, 'First return the resource');
   else
      select return_date into datereturn from return where :new.Loan_id = return.loan_id;
      if not( TRUNC((datereturn - datedue), 0) = :new.Amount_Paid) then
      raise_application_error(-20323, 'The amount being paid is unequal the loan amount');
      end if;
   end if;
end;
/
```

The trigger does the following before any fine can be paid:

- Set fine date to the system date
- Check if the resource is returned before the payment can be made
- Make sure the correct amount related to the Loan_ID is being paid

Update Fine Date:

```
select * from fine;
insert into fine_payment (LOAN_ID, AMOUNT_PAID) VALUES (1000026, 8);
 select * from fine
```

LOANID	MID	RID	FINE	ONDATE
1000026	202110	100032	8	09-DEC-20
1000030	202001	100025	2	11-DEC-20
1000033	202105	100019	1	13-DEC-20

Download CSV
3 rows selected.

1 row(s) inserted.

LOANID	MID	RID	FINE	ONDATE
1000030	202001	100025	2	11-DEC-20
1000033	202105	100019	1	13-DEC-20

Download CSV

Check if the resource is returned before the payment can be made:

select * from fine;

insert into fine_payment (LOAN_ID, AMOUNT_PAID) VALUES (1000001, 102);

LOANID	MID	RID	FINE	ONDATE
1000001	202005	100036	102	13-DEC-20
1000003	202003	100002	93	13-DEC-20
1000008	202008	100005	101	13-DEC-20
1000010	202010	100007	63	13-DEC-20
1000016	202102	100006	34	13-DEC-20
1000026	202110	100032	8	09-DEC-20
1000030	202001	100025	2	11-DEC-20
1000033	202105	100019	1	13-DEC-20

Download CSV
8 rows selected.

ORA-20321: First return the resource ORA-06512: at "SQL_GUFJPPTKOBZXSLVMGYMMKTVIW.FINE_PAY_CHECK", line 10
ORA-06512: at "SYS.DBMS_SQL", line 1721

Make sure the correct amount related to the Loan_ID is being paid:

select * from fine;

insert into return (LOAN_ID) values (1000003);

insert into fine_payment (LOAN_ID, AMOUNT_PAID) VALUES (1000003, 91);

LOANID	MID	RID	FINE	ONDATE
1000001	202005	100036	102	13-DEC-20
1000003	202003	100002	93	13-DEC-20
1000008	202008	100005	101	13-DEC-20
1000010	202010	100007	63	13-DEC-20
1000016	202102	100006	34	13-DEC-20
1000026	202110	100032	8	09-DEC-20
1000030	202001	100025	2	11-DEC-20
1000033	202105	100019	1	13-DEC-20

Download CSV
8 rows selected.

1 row(s) inserted.

ORA-20323: The amount being paid is unequal the loan amount ORA-06512: at "SQL_GUFJPPTKOBZXSLVMGYMMKTVIW.FINE_PAY_CHECK", line 14
ORA-06512: at "SYS.DBMS_SQL", line 1721

4. After Insert in Fine_Payment Table:

```
create or replace trigger update_fine
after insert on fine_payment
for each row
begin
   DELETE fine
   Where :new.Loan_ID = Fine.LoanID;
end;
/
```

When a payment is made for a loan, the trigger is made to delete the row from the fine table.

```
select * from fine;
insert into fine_payment (LOAN_ID, AMOUNT_PAID) VALUES (1000003, 93);
select * from fine;
```

LOANID	MID	RID	FINE	ONDATE
1000001	202005	100036	102	13-DEC-20
1000003	202003	100002	93	13-DEC-20
1000008	202008	100005	101	13-DEC-20
1000010	202010	100007	63	13-DEC-20
1000016	202102	100006	34	13-DEC-20
1000026	202110	100032	8	09-DEC-20
1000030	202001	100025	2	11-DEC-20
1000033	202105	100019	1	13-DEC-20

Download CSV

8 rows selected.

1 row(s) inserted.

LOANID	MID	RID	FINE	ONDATE
1000001	202005	100036	102	13-DEC-20
1000008	202008	100005	101	13-DEC-20
1000010	202010	100007	63	13-DEC-20
1000016	202102	100006	34	13-DEC-20
1000026	202110	100032	8	09-DEC-20
1000030	202001	100025	2	11-DEC-20
1000033	202105	100019	1	13-DEC-20

Data Security

Libraries are a good target for potential hackers as they are a place for open access to information. They also have fewer resources to invest in cyber protection since the services they offer are usually free of charge, so they are particularly vulnerable. Under UK law, the College Library System will have to comply with the Data Protection Act 2018, which is the UK's implementation of the General Data Protection Regulation (GDPR). This controls how personal information can be used by the library.

The Act states that information must be used transparently and for explicit purposes. The library will collect personal information so that it can identify individuals and create their accounts, such as Member_ID, Member_Name and Member_Email. This information will only be obtained directly from the member or when the member uses the library services and just the relevant information will be taken. No sensitive information will be collected unless absolutely necessary, for example, any access requirements members may have to comply with legal obligations under equality legislation.

Additionally, the Act maintains that the information should be processed in a manner that guarantees appropriate security. Access controls will be set up to ensure that personal information is only be viewed by relevant library staff. This will limit the number of people that will be able to view, modify or delete the data.

Access to personal data will only be granted to staff when it is needed for legitimate operational purposes. Staff will have their privileges revoked once they are no longer working at the library. Database administrators will be responsible for setting up employee access and control who has access to what. For instance, an employee who has permission to see certain pieces of information may be denied the ability to make any changes to them.

The separation of administrator and user powers with the segregation of duties will make it harder for internal staff to carry out fraud or theft. Limiting the power of user accounts will also make it more difficult for hackers to take total control of a database. If unauthorised users gain access to the database, they can cause several problems such as holding data hostage for ransom, as was the case with the NHS in 2017.

The MEMBER table will use pseudonymisation to replace identifying information with artificial identifiers to protect data in the event of a hack. The data will be encrypted as well using Oracle's Transparent Data Encryption so that only approved users have access to it. This will also mitigate insider threats.

Backup and recovery procedures will be in place to protect the database against data loss and for reconstruction in the unfortunate event that data is lost. Database administrators (DBA) will be in charge of deciding which backup mechanisms to use and how frequently the data is backed up. The DBA will be responsible for periodically checking the backup and recovery processes work by recreating the database from a backup and ensuring that it is fully functional.

Furthermore, the Data Protection Act 2018 also states that information cannot be kept for longer than is necessary. Therefore, personal information will be deleted within 90 days of membership expiry. This will be accomplished using an AFTER UPDATE trigger. The value of Member_ID can be kept as it is not personal, Member_Name and Member_Email will be set to a default value. Keeping Member_ID as it is will avoid cascading effects, so the loan details will not have to be altered. The data will be removed from any backups as well. A BEFORE UPDATE trigger will be implemented to ensure that a member's information is not deleted if they have any unpaid fines or outstanding loans.

Bibliography

GOV.UK. *Data Protection*. [Online] Available from: https://www.gov.uk/data-protection [Accessed 3 December 2020].

Kaspersky. (2020). *What Is Wannacry Ransomware?*. [Online] Available from: https://www.kaspersky.co.uk/resource-center/threats/ransomware-wannacry [Accessed 4 December 2020].

Lemahieu, W. (2018). *Principles of Database Management: The Practical Guide to Storing, Managing and Analyzing Big and Small Data*. Cambridge: Cambridge University Press.

Stockman, A. (2020). *Coursework 2: Database Implementation*. Available from: https://qmplus.qmul.ac.uk/course/view.php?id=15472 [Accessed 16 November 2020].