**University of Derby Online Learning**

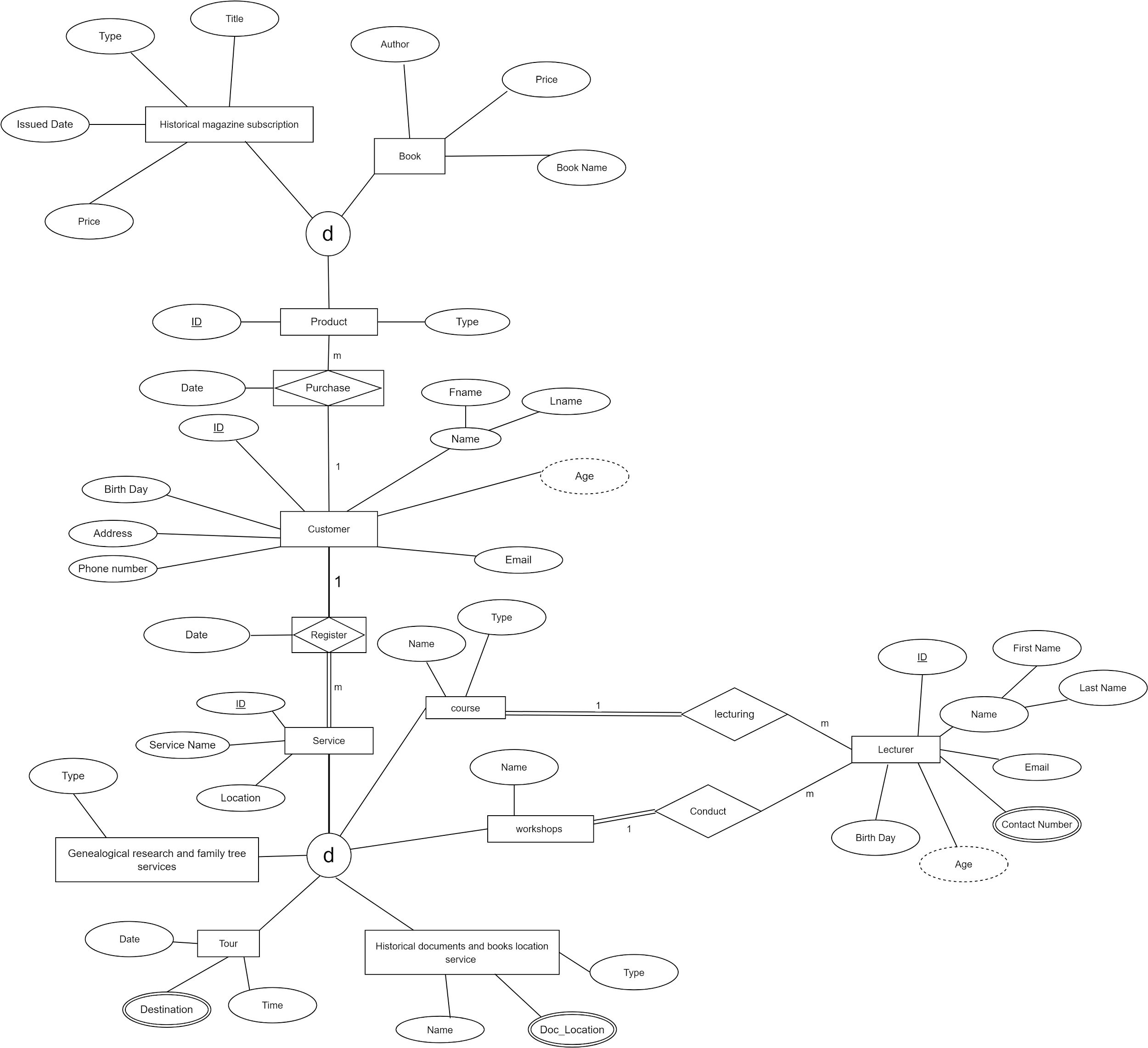
**6CC546 - Database Fundamentals**

**September 2022**

Full Name:

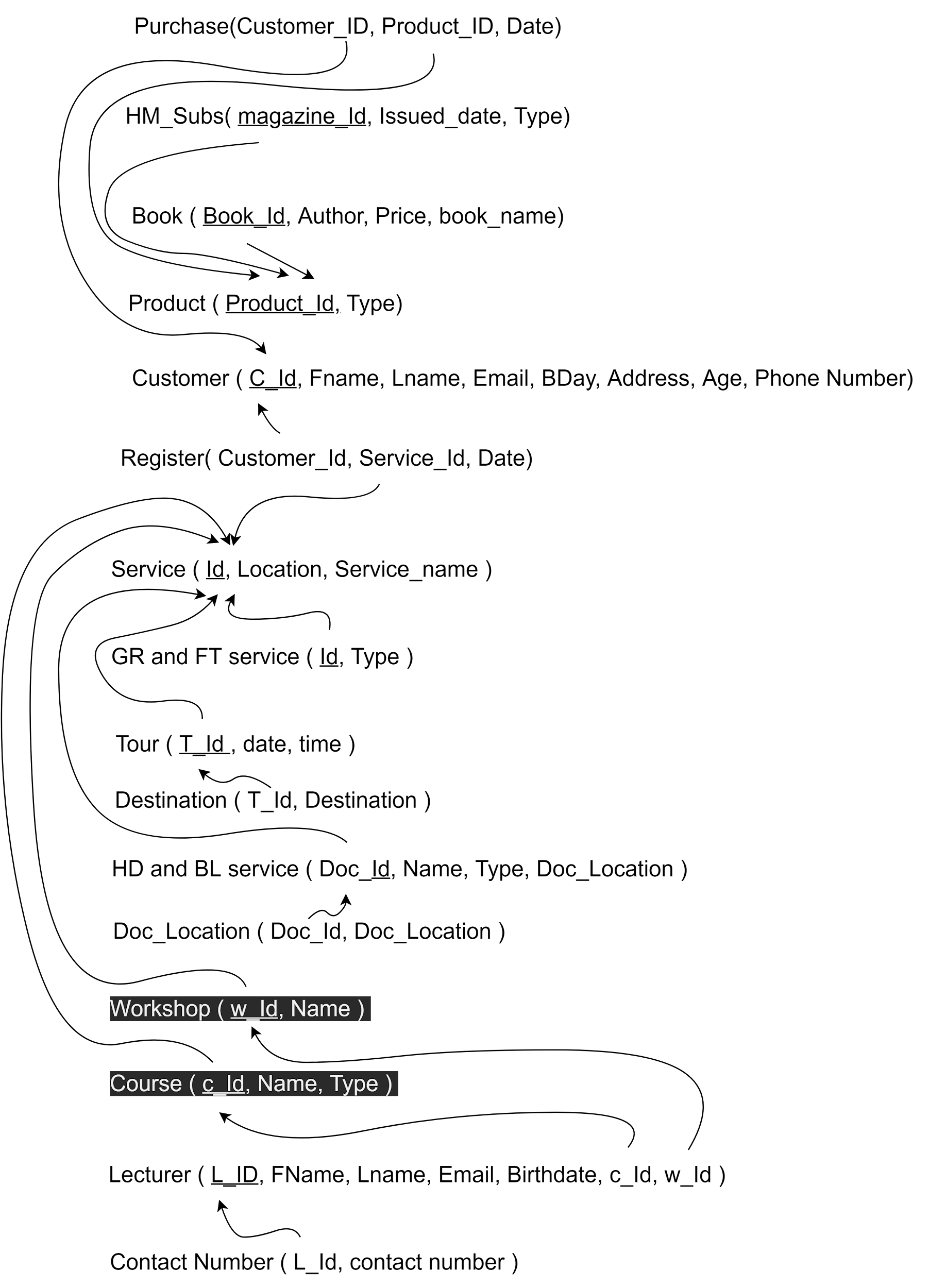
Student ID:

**1. Entity-Relationship Model**



* Assumptions used while designing the ER diagram
  + Lecturers conduct the courses and workshops
  + Courses could be virtual or on premisses
  + There are multiple branches/locations to the company
  + There could be multiple destinations to a tour
  + There could be multiple document locations to a historical document or book
  + Customer can purchase multiple books, magazines and register multiple services
* Relationships
  + Customer purchase product - one to many
  + Customer register for service - one to many
  + Lecturer lecturing for the course - Many to one
  + Lecturer conduct workshops - Many to one

**2. Boyce-Codd Normal Form (BCNF)**



**1NF**

In the first normal form,

* Each table cell should contain a single value.
* Each record needs to be unique.

There were multi-value attributes in the ER diagram such as destinations in the tour entity, Doc\_location in the historical documents and book location service entity, and Contact number in the lecturer entity. Within the first normal form, we separate those multivalued attributes in column

**2NF**

* Rule 1- Be in 1NF
* Rule 2- Single Column Primary Key that does not functionally dependant on any subset of candidate key relation

Here we separate the attribute tables of,

Lecturer entity to lecturer and contact number tables

Each service to separate tables

Historical documents and book location entity to HD and BL service and Doc location tables

Tour entity to tour and destination tables

**3NF and BCNF**

* Rule 1- Be in 2NF
* Rule 2- Has no transitive functional dependencies

To achieve the third normal form,

Created separate Historical Magazine subscription, book and product tables.

Separate tables for customers and register tables for customers get to register for services

**3. Final database design**

**SQL Code to implement the required tables and relationships**

#delete database if exists

| DROP DATABASE IF EXISTS History;  #create the database CREATE DATABASE History;  USE History;  #create a table for product CREATE TABLE Product(  Product\_id INT AUTO\_INCREMENT NOT NULL PRIMARY KEY,  Type VARCHAR(255) );  #create a table for books CREATE TABLE Book(  Book\_id INT NOT NULL,  Author VARCHAR(255),  Price VARCHAR(255),  Book\_name VARCHAR(255),  FOREIGN KEY (Book\_id) REFERENCES Product(Product\_id) );  #create a table for historical magazines CREATE TABLE HM\_Subs(  Mag\_id INT NOT NULL,  Title VARCHAR(255),  Price VARCHAR(255),  issue\_date DATE,  Type VARCHAR(255),  FOREIGN KEY (Mag\_id) REFERENCES Product(Product\_id) );  #create a table for services CREATE TABLE Service (  id INT NOT NULL AUTO\_INCREMENT,  service\_name VARCHAR(255),  location VARCHAR(255),  PRIMARY KEY(id) );  #create table for customers  CREATE TABLE Customer(  C\_id INT NOT NULL AUTO\_INCREMENT,  Fname VARCHAR(255),  Lname VARCHAR(255),  Email VARCHAR(255),  Bday DATE,  Address VARCHAR(255),  Phone\_number VARCHAR(255),  PRIMARY KEY(C\_id) );  #create table for genealogical research and family tree service CREATE TABLE GR\_and\_FT (  serv\_id INT NOT NULL,  type VARCHAR(255),  FOREIGN KEY (serv\_id) REFERENCES Service(id) );  #create table for tours CREATE TABLE Tour(  T\_id INT NOT NULL,  T\_date DATE,  T\_time TIME,  FOREIGN KEY (T\_id) REFERENCES service(id) );  #create table for tour destinations CREATE TABLE Destination(  T\_id INT NOT NULL,  Destination VARCHAR(255) NOT NULL,  FOREIGN KEY (T\_id) REFERENCES service(id) );  #create table for historical documents and book location service CREATE TABLE HD\_and\_BL(  D\_id INT NOT NULL,  Name VARCHAR(255) NOT NULL,  Type VARCHAR(255),  Doc\_Location VARCHAR(255),  FOREIGN KEY (D\_id) REFERENCES service(id) );  #table to store document locations CREATE TABLE Doc\_Location(  D\_id INT NOT NULL,  Doc\_Location VARCHAR(255) NOT NULL,  FOREIGN KEY (D\_id) REFERENCES service(id) );  #Table for workshops CREATE TABLE Workshop(  W\_id INT NOT NULL,  Name VARCHAR(255),  FOREIGN KEY (W\_id) REFERENCES service(id) );  #Talbe for courses CREATE TABLE Course(  C\_id INT NOT NULL,  Name VARCHAR(255),  Type VARCHAR(255),  FOREIGN KEY (C\_id) REFERENCES service(id) );  #Table for store lecturer details CREATE TABLE Lecturer(  L\_id INT NOT NULL PRIMARY KEY AUTO\_INCREMENT,  FName VARCHAR(255),  LName VARCHAR(255),  Email VARCHAR(255),  B\_day DATE,  C\_id INT NOT NULL,  W\_id INT NOT NULL,  FOREIGN KEY (C\_id) REFERENCES service(id),  FOREIGN KEY (W\_id) REFERENCES service(id)  );  #table to store lecturer contact numbers CREATE TABLE Contact\_number(  L\_id INT NOT NULL,  contact\_number VARCHAR(255),  FOREIGN KEY (L\_id) REFERENCES Lecturer(L\_id) );  #create table to record product purchases CREATE TABLE Purchase(  customer\_id INT NOT NULL,  product\_id INT NOT NULL,  date DATE,  FOREIGN KEY (customer\_id) REFERENCES customer(C\_id),  FOREIGN KEY (product\_id) REFERENCES product(Product\_id) );  #table for register customers to the services CREATE TABLE Register(  customer\_id INT NOT NULL,  service\_id INT NOT NULL,  date DATE,  FOREIGN KEY (customer\_id) REFERENCES customer(C\_id),  FOREIGN KEY (service\_id) REFERENCES service(id) ); |
| --- |

**Populate the database**

Here we add the sample data and populate the database

| USE history;  INSERT INTO Service(service\_name, location)  VALUE  ("tour", "UK"),  ("tour", "India"),  ("GR and FT", "virtual"),  ("GR and FT", "Colombo"),  ("HD and BL", "Colombo"),  ("HD and BL", "Kandy"),  ("workshop", "Colombo"),  ("workshop", "Kandy"),  ("course", "Colombo"),  ("course", "Kandy")  ;    INSERT INTO Tour(T\_id,T\_date, T\_time)   VALUE  (1,"2022/02/10","03:00:00"),  (2,"2022/03/10","03:00:00");      INSERT INTO gr\_and\_ft(serv\_id, type)  VALUE   (3,"GR"),  (4,"both");    INSERT INTO hd\_and\_bl(D\_id, Name, Type, Doc\_Location)  VALUES  (5,"The Constitution", "book", "Yokshire"),  (6,"Magna Carta", "document", "Bristol");   INSERT INTO Workshop(W\_id, Name) VALUES (7,"Portals to the Past | Inspiring history workshops for schools"), (8,"World History Workshop");  INSERT INTO Course(C\_id, Name, Type) VALUES (9,"Forensic Archaeology and Anthropology","On Premises"), (10,"Genealogy: Researching Your Family Tree","On Premises");   INSERT INTO Product(Type)  VALUES ("book"),("book"),("book"),("magazine"),("magazine");  INSERT INTO Book(Book\_id,Book\_name,Author,Price)  VALUES (1,"The Ottomans: Khans, Caesars and Caliphs","Marc David Baer","800"), (2,"The First World War","Michael Howard","900"), (3,"The Greeks and the Irrational","E R Dodds","750");  INSERT INTO HM\_subs(Mag\_id, Title, Price, issue\_date, Type) VALUES (4, "England History", 200, "2022/07/10", "Online"), (5, "Step Into Past", 300, "2022/08/10", "Online");  INSERT INTO Customer(Fname,Lname,Email,Bday,Address,Phone\_number) VALUES  ("Sanath","Kumara","sanath@gmail.com","1988/12/15","26/8,Katubadda, Moratuwa","0774589641"),  ("Nadun","Silva","nadun@gmail.com","1998/12/25","26/7,Weera Mavatha, Gampaha","0774589641"),  ("Nimesh","Silva","nimesh@gmail.com","1999/10/25","25/7,Sahana Mavatha, Gampaha","0785689641");  INSERT INTO Destination(T\_id,Destination) VALUES (1,"Edinburgh"), (1,"Ancient Stonehenge and Medieval Salisbury"), (1,"Idyllic England: The Cotswolds"), (2,"Shimla, Himachal Pradesh"), (2,"Nainital, Uttarakhand");  INSERT INTO Lecturer(FName, LName, Email, B\_day, C\_id, W\_id) VALUES ("John","Edmon","john@gmail.com","1985/05/12", 9,7), ("Anna","Silvia","anna@gmail.com","1975/02/16", 10,7), ("Semon","Silva","semon@gmail.com","1986/08/26", 9,8);  INSERT INTO purchase(customer\_id, product\_id, date) VALUES (1,2,"2022/10/18"), (2,3,"2022/10/18");  INSERT INTO register(customer\_id, service\_id, date) VALUES (1,3,"2022/10/18"), (2,4,"2022/10/17"), (1,9,"2022/10/17"), (3,10,"2022/10/17"); |
| --- |

**3. Database Queries**

| #basic query for select customer Sanath SELECT \* FROM Customer WHERE Fname = "Sanath";  #Renaming column of hd\_and\_bl ALTER TABLE Course CHANGE Name Course\_name VARCHAR(255);  #join 3 tables to find the customers registered for courses SELECT course.C\_id,Customer.Fname, Customer.Lname, Customer.Email,Customer.Address,Customer.Phone\_number, course.Course\_name, Course.Type FROM customer INNER JOIN register ON register.customer\_id = Customer.C\_id INNER JOIN course ON course.C\_id = register.service\_id;  #join 4 tables and subqueries  SELECT destination.T\_id,Customer.Fname, Customer.Email,Customer.Address,Customer.Phone\_number,destination.Destination, service.location FROM ((destination INNER JOIN register ON destination.T\_id = register.service\_id) INNER JOIN Customer ON register.customer\_id = Customer.C\_id) INNER JOIN service ON service.id = register.service\_id;   #order by ASC of books  SELECT \* FROM Book ORDER BY Price ASC;    #count - customer count SELECT COUNT(C\_id) FROM Customer;  #Max - Book that available for maximum price SELECT MAX(Price),Book\_name,Author FROM Book;  #union and SQL Aliases - All the courses and workshop details SELECT W\_id AS id, Name FROM Workshop UNION ALL SELECT C\_id, Course\_name AS id FROM Course;    #lecture details SELECT CONCAT(FName," ", Lname) AS Lecturer\_name, Email  FROM Lecturer;  #The customers who got John as him/her lecturer in course SELECT CONCAT(Customer.FName," ",Customer.Lname) AS Customer\_name, Lecturer.FName, Lecturer.Email AS Lecturer\_mail, Course.Course\_name FROM( Customer INNER JOIN Lecturer ON Lecturer.FName = "John") INNER JOIN register ON register.customer\_id = Customer.C\_id INNER JOIN Course ON Course.C\_id = register.Service\_id;   #select workshop with lecturer and location SELECT service.id, workshop.Name as name,lecturer.L\_id, CONCAT(lecturer.FName,' ',lecturer.LName) AS lecturer, service.location FROM service INNER JOIN Lecturer ON Lecturer.W\_id = service.id INNER JOIN workshop ON workshop.W\_id = service.id;  #course with locaion SELECT service.id, course.Course\_name as name, course.Type, service.location FROM service JOIN course ON course.C\_id = service.id;  #Select purchase details SELECT product.Product\_id,purchase.date , CONCAT(customer.Fname, ' ' , customer.Lname) AS customer, customer.Email, product.Type FROM purchase JOIN customer ON customer.C\_id = purchase.customer\_id JOIN product ON product.Product\_id = purchase.product\_id;  #select all products SELECT Mag\_id AS ID,Title,Price FROM hm\_subs UNION SELECT Book\_id, Book\_name, Price FROM book; |
| --- |
|  |

**5. Optimize the Database**

| use history; #basic query CREATE INDEX Customer\_asc ON Customer(Fname ASC); SELECT CONCAT(Fname, " ", Lname) AS Name, Email,Phone\_number FROM Customer WHERE Fname = "Sanath";  #join 3 tables to find the customers registered for courses  SELECT co.C\_id,cu.Fname, cu.Lname, cu.Email,cu.Address,cu.Phone\_number, co.Course\_name, co.Type FROM customer cu INNER JOIN register ON register.customer\_id = cu.C\_id INNER JOIN course co ON co.C\_id = register.service\_id;    #join 4 tables and subqueries  SELECT d.T\_id,c.Fname, c.Email,c.Address,c.Phone\_number,d.Destination, s.location FROM ((destination d INNER JOIN register ON d.T\_id = register.service\_id) INNER JOIN Customer c ON register.customer\_id = c.C\_id) INNER JOIN service s ON s.id = register.service\_id;   #order by ASC CREATE INDEX Book\_price\_asc ON Book(Price ASC); SELECT Book\_name, Price FROM Book ORDER BY Price ASC;    #Max SELECT MAX(Price),Book\_name,Author FROM Book; #union and SQL Aliases SELECT W\_id AS id, Name FROM Workshop UNION ALL SELECT C\_id, Course\_name AS id FROM Course;  #lecture details SELECT CONCAT(FName," ",Lname) AS Lecturer\_name, Email  FROM Lecturer;   #The customers who got John as him/her lecturer in course CREATE INDEX Lecturer\_names ON Lecturer(Fname ASC, Lname ASC); SELECT CONCAT(cu.FName," ",cu.Lname) AS Customer\_name, l.FName, l.Email AS Lecturer\_mail, co.Course\_name FROM( Customer cu INNER JOIN Lecturer l ON l.FName = "John") INNER JOIN register r ON r.customer\_id = cu.C\_id INNER JOIN Course co ON co.C\_id = r.Service\_id;   #select workshop with lecturer and location SELECT s.id, w.Name as name,l.L\_id, CONCAT(l.FName,' ',l.LName) AS lecturer, s.location FROM service s INNER JOIN Lecturer l ON l.W\_id = s.id INNER JOIN workshop w ON w.W\_id = s.id;  #course with locaion SELECT s.id, c.Course\_name as name, c.Type, s.location FROM service s JOIN course c ON c.C\_id = s.id;  #Select purchase details SELECT pr.Product\_id,p.date , CONCAT(c.Fname, ' ' , c.Lname) AS customer, c.Email, pr.Type FROM purchase p JOIN customer c ON c.C\_id = p.customer\_id JOIN product pr ON pr.Product\_id = p.product\_id;  #select all products SELECT Mag\_id AS ID,Title,Price FROM hm\_subs UNION SELECT Book\_id, Book\_name, Price FROM book; |
| --- |

* To optimise the queries
  + Created ascending indexes to retrieve data faster
  + Removed unused indexes
  + Used SELECT fields instead of using SELECT \*
  + Avoided SELECT DISTINCT
  + Created joins with INNER JOIN (not WHERE)
  + Used Aliases

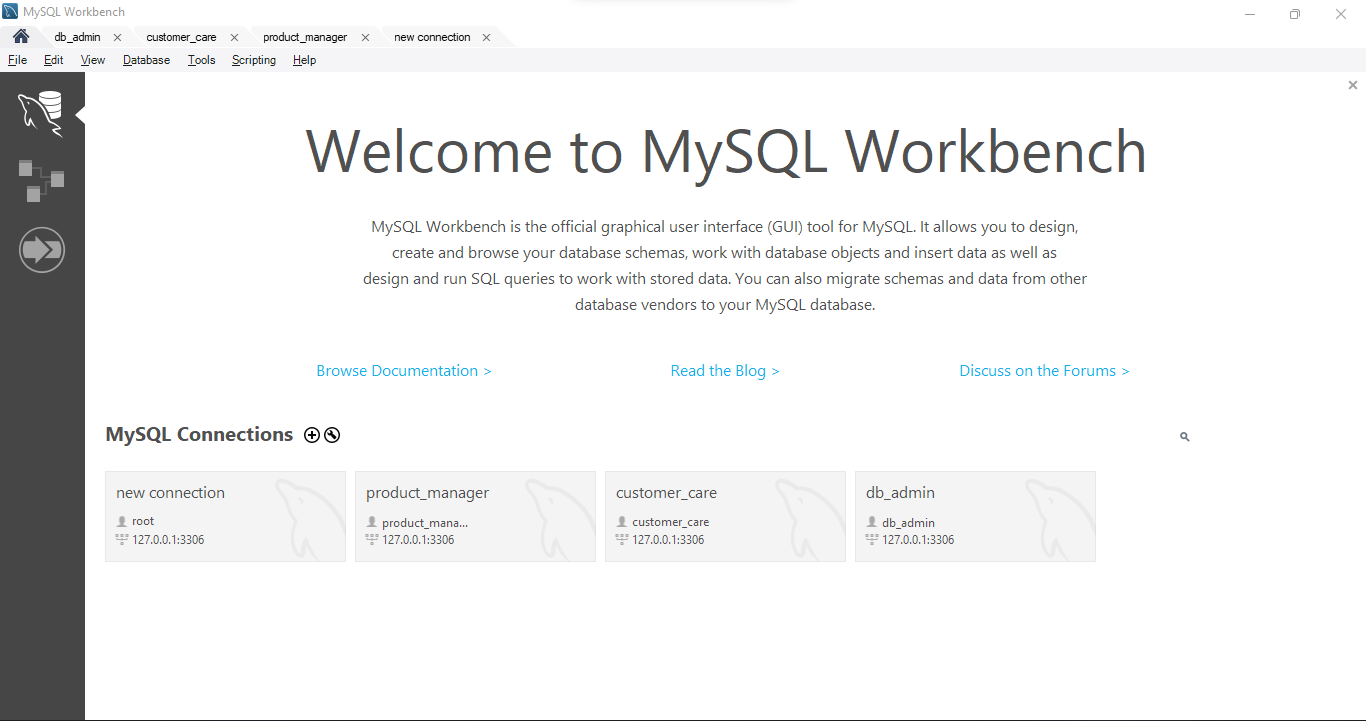
Using the above-mentioned techniques,

* speed up searches of your database by allowing MySQL to organize your data in the best way for different queries
* Select and retrieve only relevant fields
* Use less effort to execute the queries
* Use less time to retrieve the relevant records

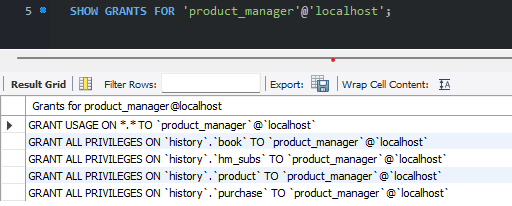
**6. Secure the Database**

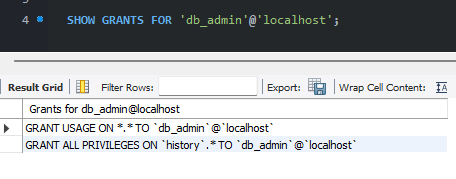
| CREATE USER 'product\_manager'@'localhost' IDENTIFIED BY 'password'; CREATE USER 'customer\_care'@'localhost' IDENTIFIED BY 'password'; CREATE USER 'db\_admin'@'localhost' IDENTIFIED BY 'password';  GRANT ALL PRIVILEGES ON History.product TO 'product\_manager'@'localhost'; GRANT ALL PRIVILEGES ON History.purchase TO 'product\_manager'@'localhost'; GRANT ALL PRIVILEGES ON History.book TO 'product\_manager'@'localhost'; GRANT ALL PRIVILEGES ON History.hm\_subs TO 'product\_manager'@'localhost';   GRANT ALL PRIVILEGES ON History.customer TO 'customer\_care'@'localhost';  GRANT ALL PRIVILEGES ON History.\* TO 'db\_admin'@'localhost'; |
| --- |

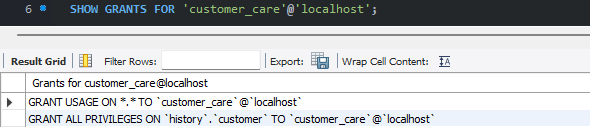
* Table access for each user



* Permissions granted for each user

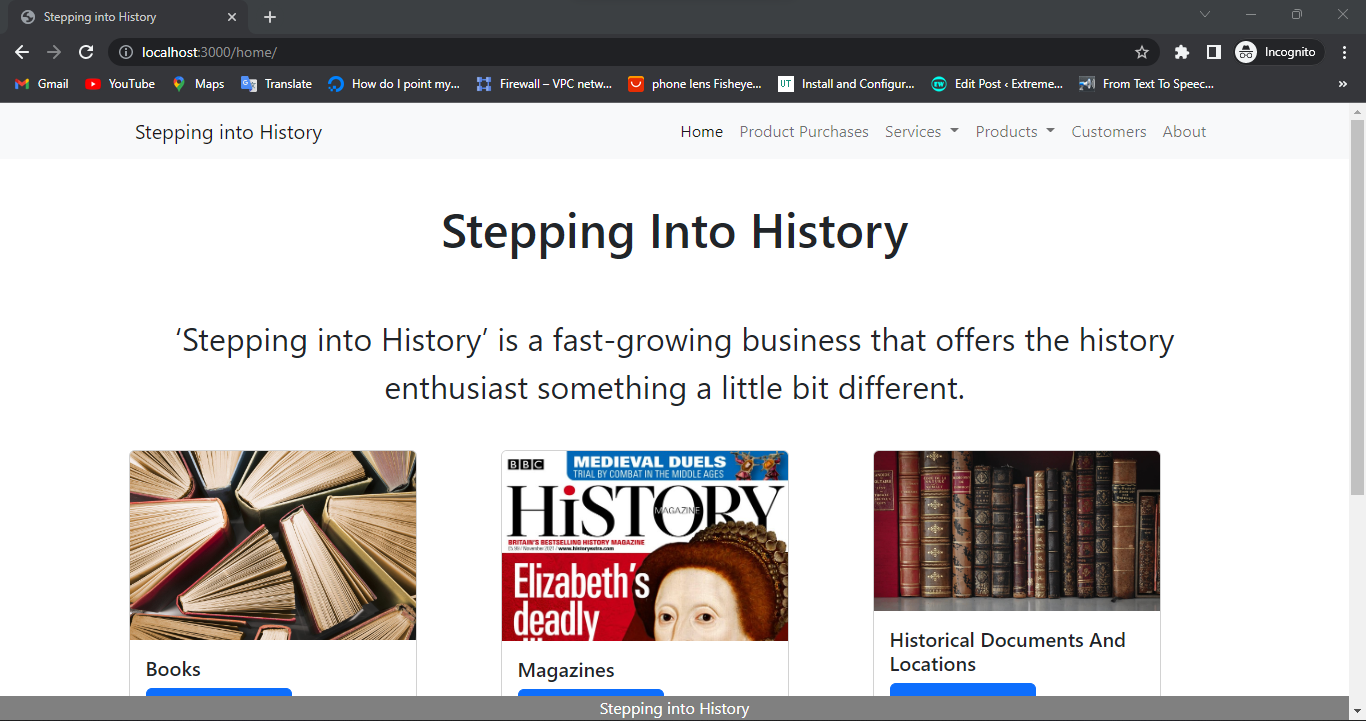


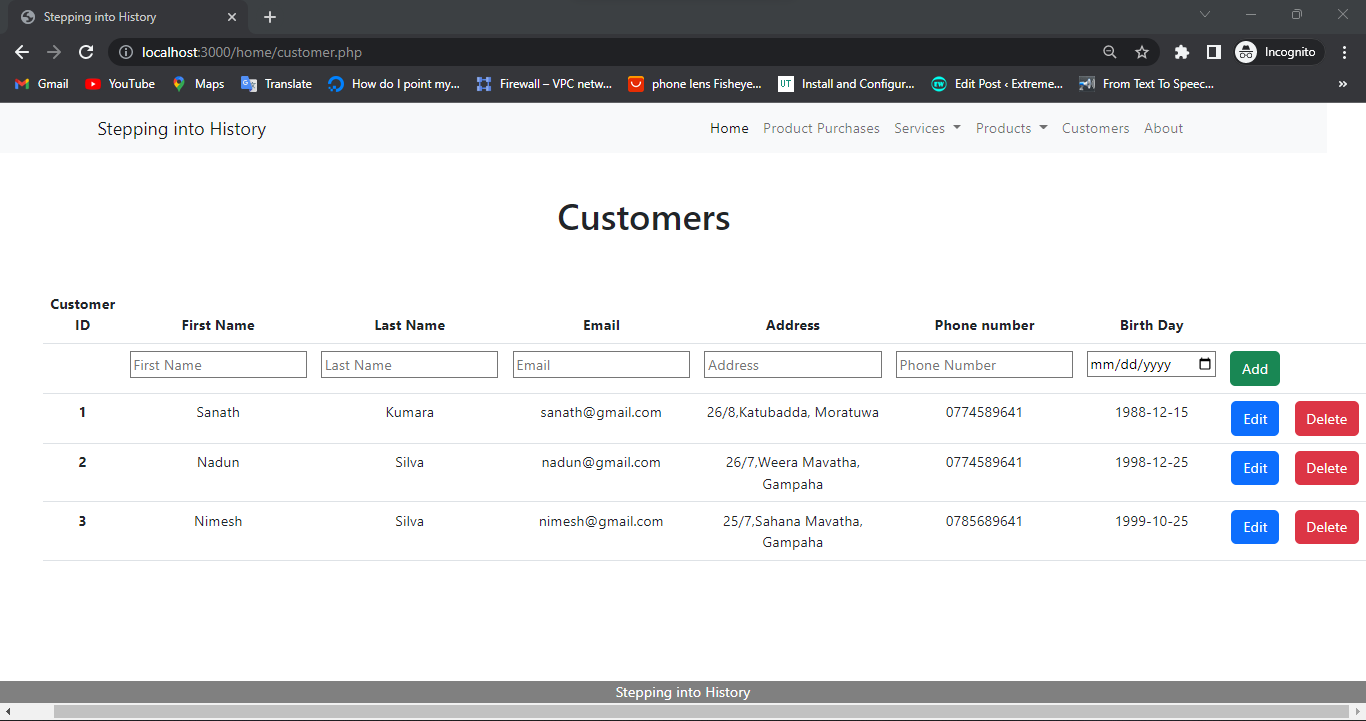


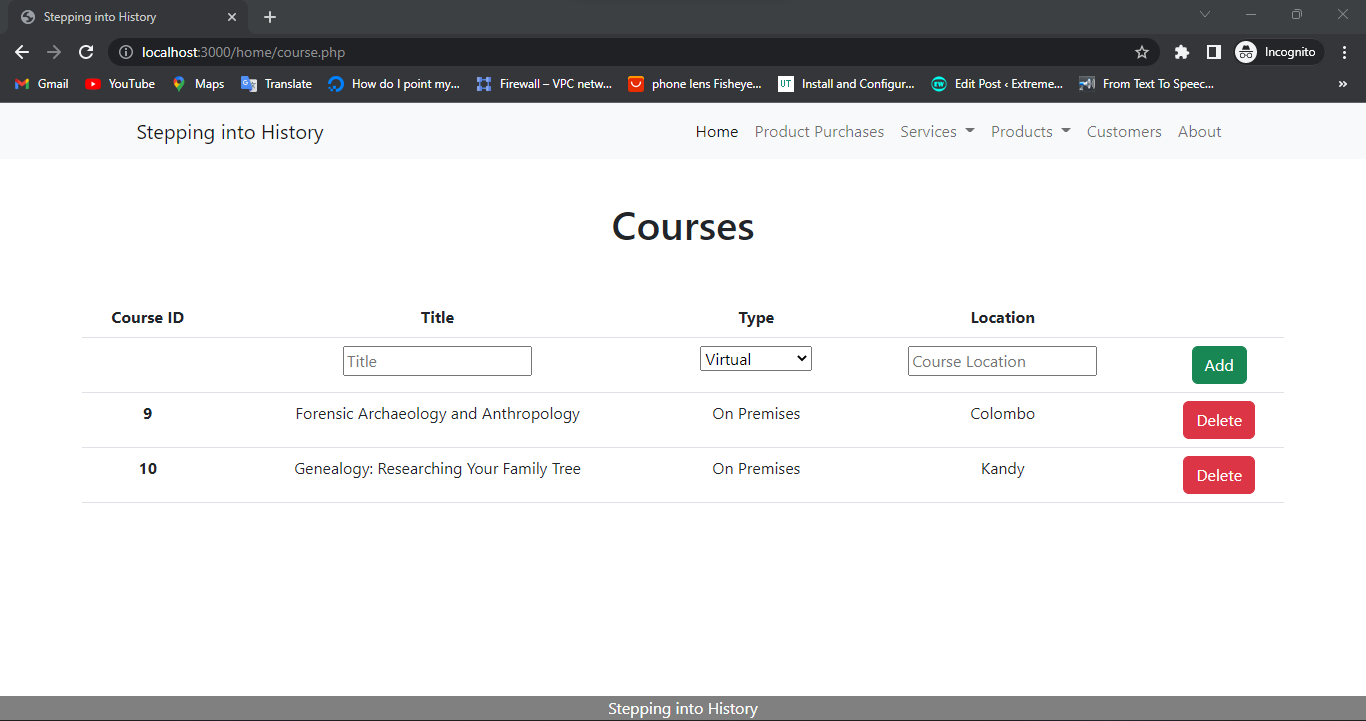


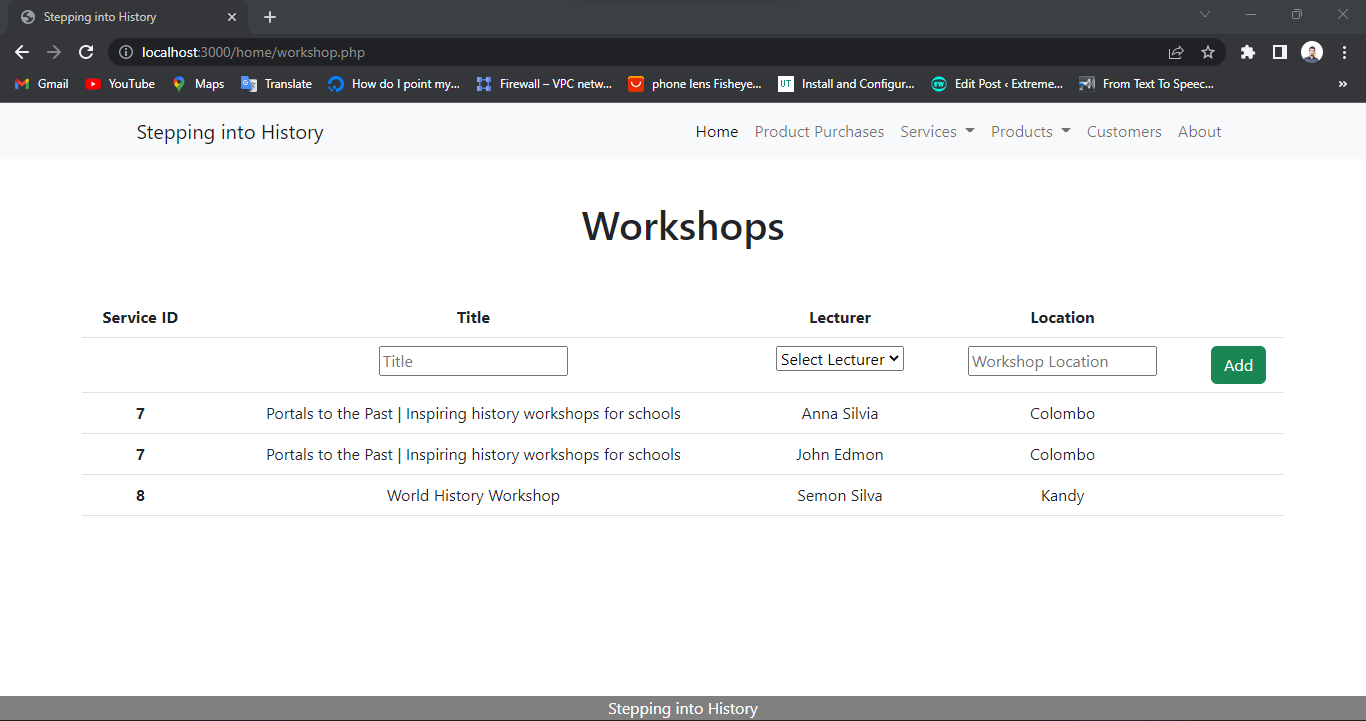
**7.Building a Web Interface**

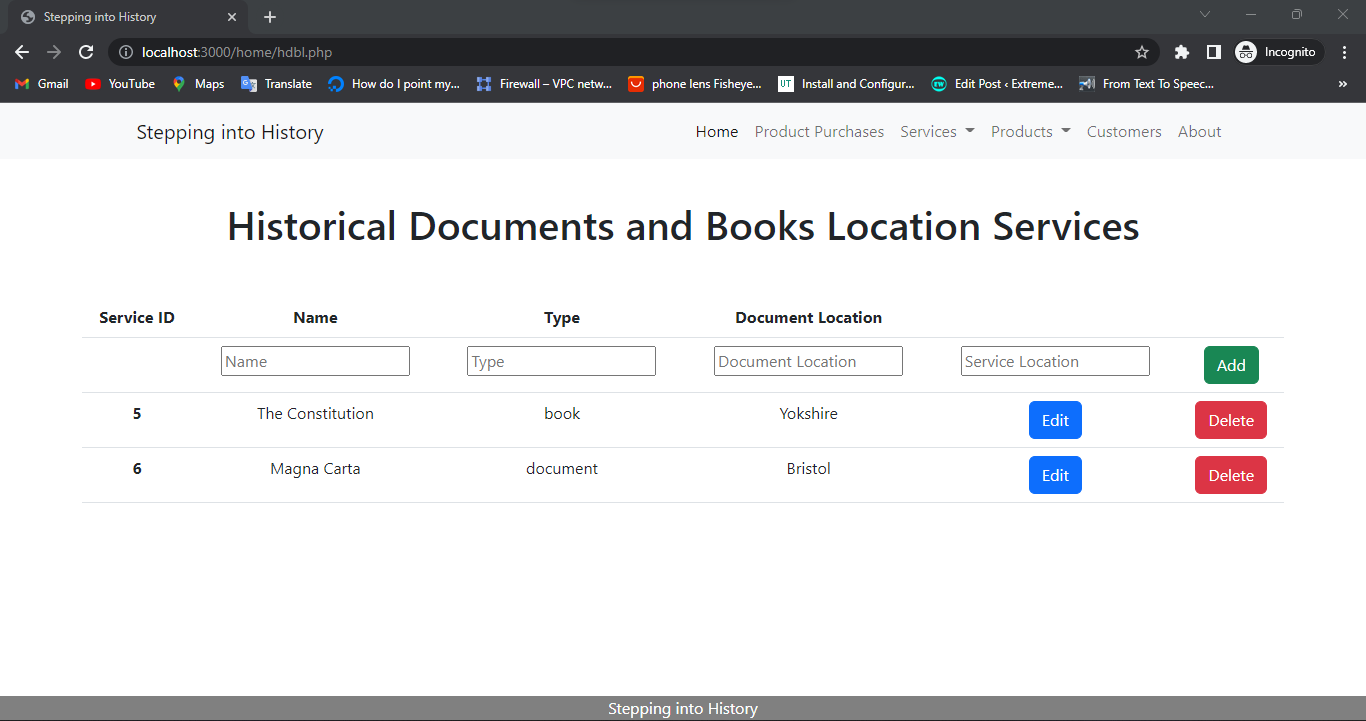
* To build the web interface here I used HTML, CSS, JavaScript and Bootstrap for frontend and PHP with MySQL for the backend.
* Implemented Add, Update, Delete and Display functions

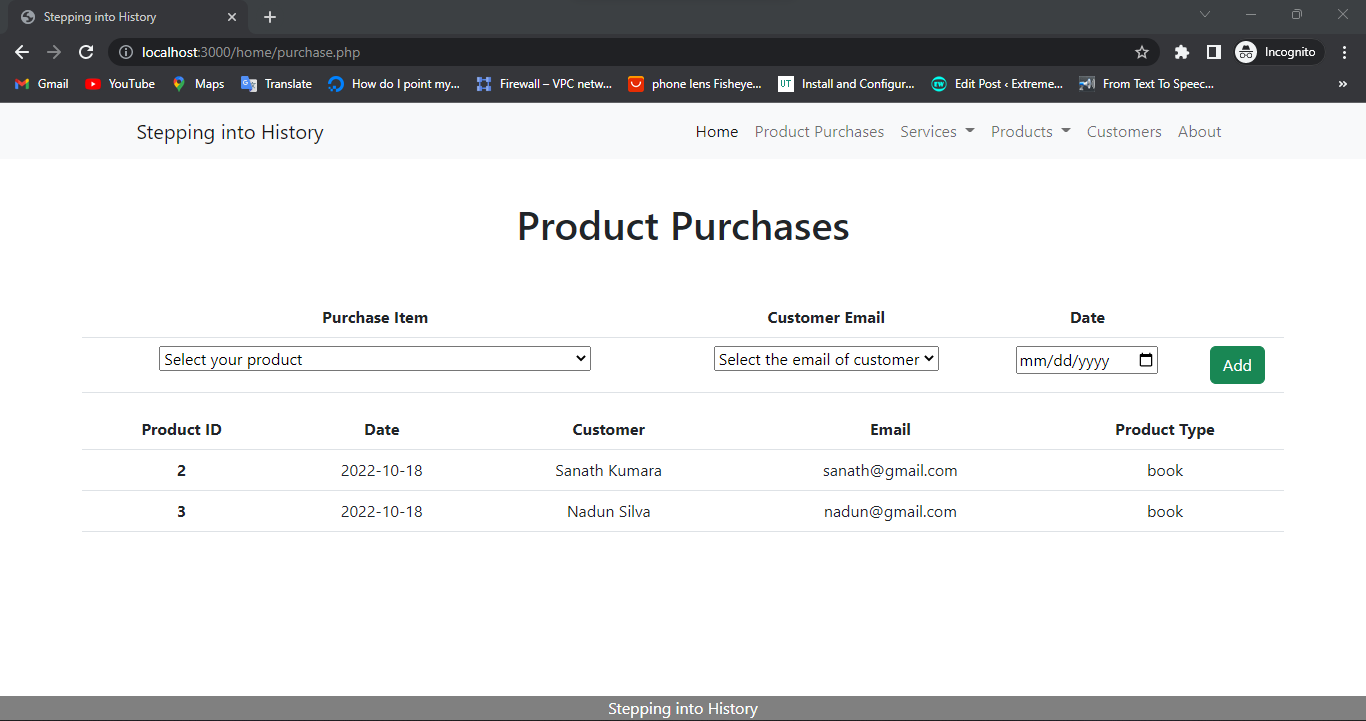


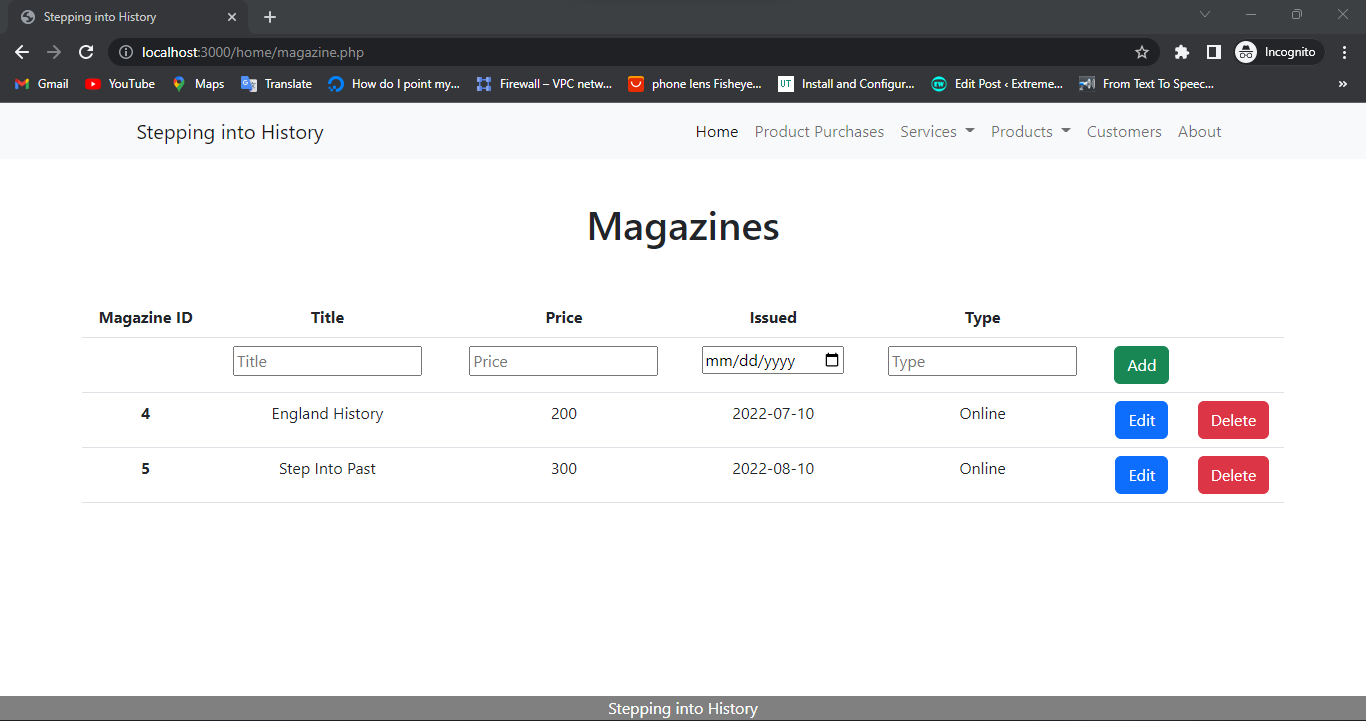


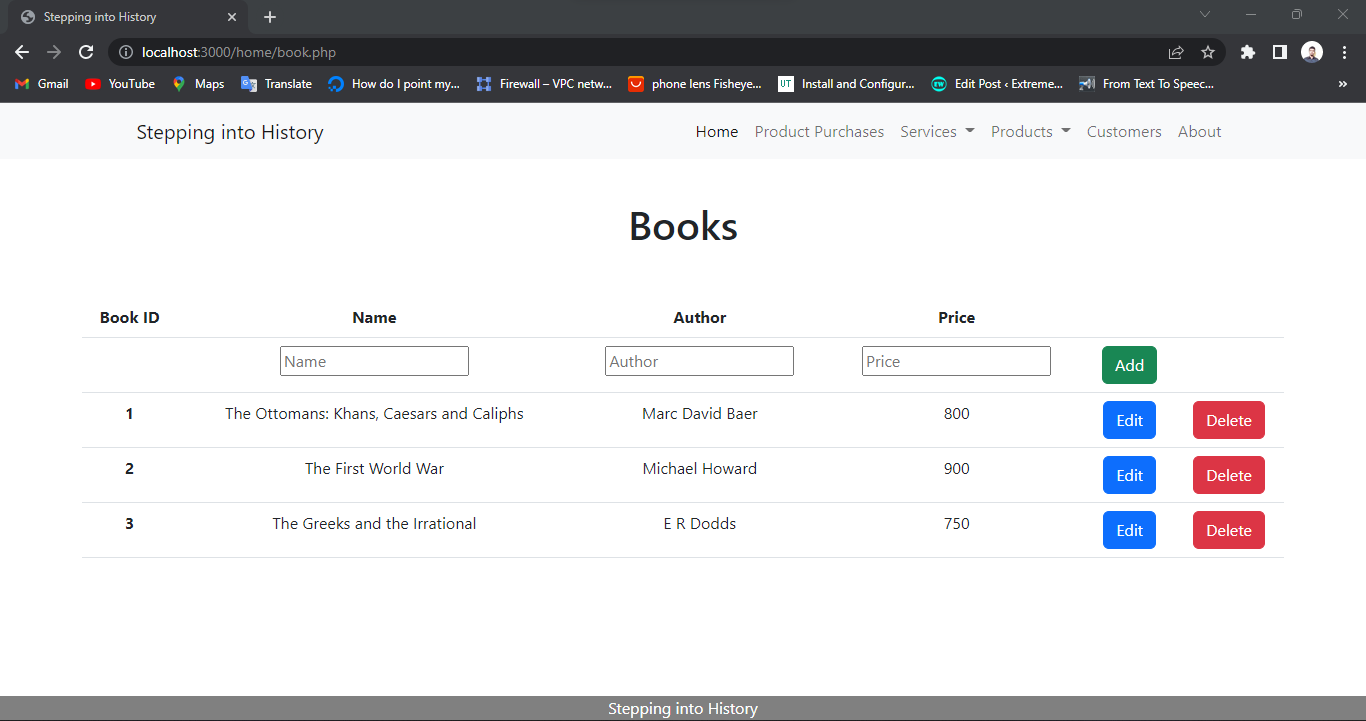


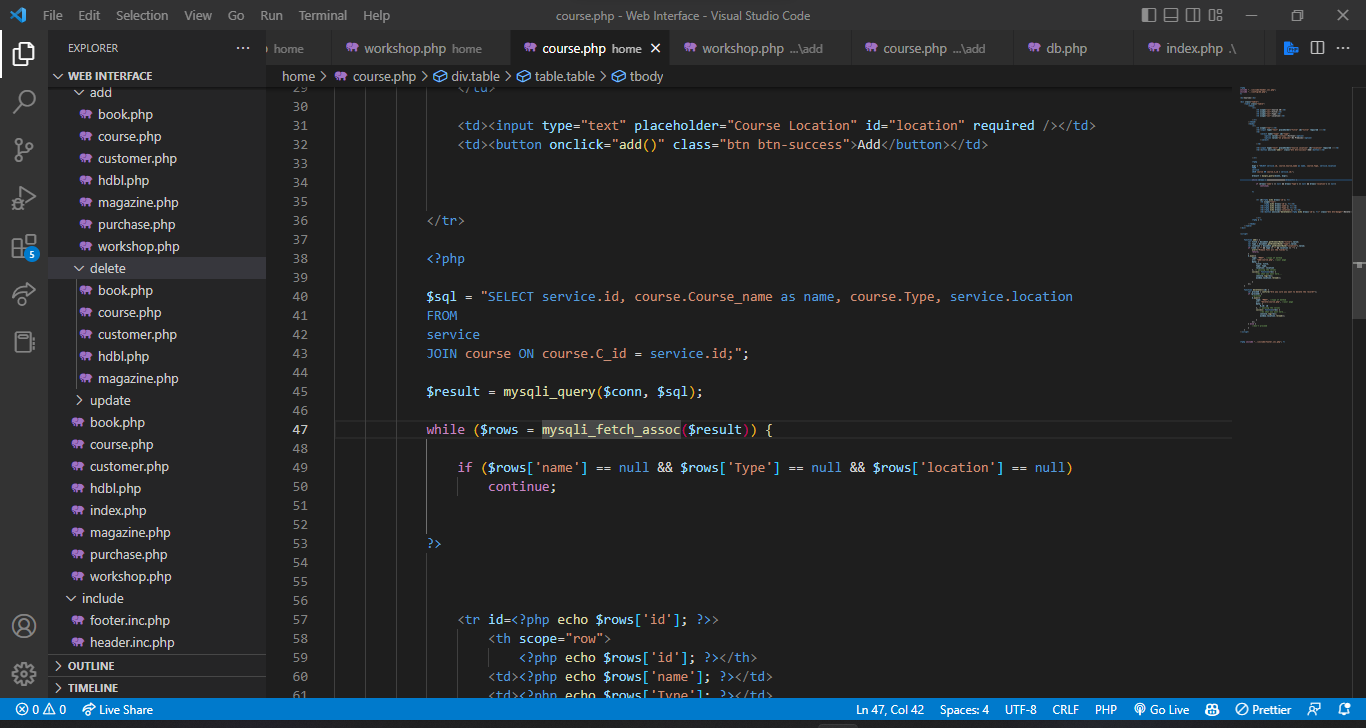












* All the codes and related files are attached to the combined zip file.
* To demonstrate the web interface - use XAMMP or apache web server with MySQL

**8. References**

contributors, M.O., Jacob Thornton, and Bootstrap (n.d.). *Get started with Bootstrap*. [online] getbootstrap.com. Available at: https://getbootstrap.com/docs/5.2/getting-started/introduction/.

Sisense. (2019). *8 Ways to Fine-Tune Your SQL Queries (for Production Databases) | Sisense*. [online] Available at: https://www.sisense.com/blog/8-ways-fine-tune-sql-queries-production-databases/.

W3Schools (2019a). *PHP Tutorial*. [online] W3schools.com. Available at: https://www.w3schools.com/php/default.asp.

W3Schools (2019b). *SQL Tutorial*. [online] W3schools.com. Available at: https://www.w3schools.com/sql/default.asp.

w3Schools (2019). *JavaScript Tutorial*. [online] W3schools.com. Available at: https://www.w3schools.com/js/default.asp.