DATABASE DESIGN REPORT

Roshan Sreekanth R00170592 SDH2-C

Table of Contents

Project Description

The project is a Flight Management system that allows for the following functions:

1. Book a ticket
2. Cancel a ticket
3. Add a new flight
4. View all flights
5. Change flight details
6. Cancel a flight

The first two options are customer-centric, and the rest are administrator-centric options.

The Java front-end interacts with an SQL Database. The user interacts with the program using a console.

Creates and Inserts

```
CREATE TABLE `customer` (
  `customerId` INT AUTO_INCREMENT NOT NULL,
  `password` VARCHAR(45) NOT NULL,
  `firstName` VARCHAR(45) NOT NULL,
  `lastName` VARCHAR(45) NOT NULL,
  `email` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`customerId`),
  UNIQUE INDEX `customerId_UNIQUE` (`customerId` ASC),
  UNIQUE INDEX `email_UNIQUE` (`email` ASC));
```

```
CREATE TABLE `carrier` (
  `carrierName` VARCHAR(40) NOT NULL,
  PRIMARY KEY (`carrierName`),
  UNIQUE INDEX `carrierName_UNIQUE` (`carrierName` ASC));

CREATE TABLE `airport` (
  `airportCode` VARCHAR(3) NOT NULL,
  `name` VARCHAR(50) NOT NULL,
  `city` VARCHAR(26) NOT NULL,
  PRIMARY KEY (`airportCode`),
  UNIQUE INDEX `airportCode_UNIQUE` (`airportCode` ASC),
  UNIQUE INDEX `name_UNIQUE` (`name` ASC));

CREATE TABLE `flight` (
  `flightNo` INT AUTO_INCREMENT NOT NULL,
  `carrierName` VARCHAR(40) NOT NULL,
  `portOfCall` VARCHAR(3) NOT NULL,
  `portOfDestination` VARCHAR(3) NOT NULL,
  `arrivalTime` TIME(0) NOT NULL,
  `departureTime` TIME(0) NOT NULL,
  `travelDate` DATE NOT NULL,
  `basePrice` DECIMAL(6,2) NOT NULL,
  PRIMARY KEY (`flightNo`),
  UNIQUE INDEX `flightNo_UNIQUE` (`flightNo` ASC),
  INDEX `carrierName_idx` (`carrierName` ASC),
```

```
  INDEX `portOfCall_idx` (`portOfCall` ASC),
  INDEX `portOfDestination_idx` (`portOfDestination` ASC),
  CONSTRAINT `carrierName`
        FOREIGN KEY (`carrierName`)
        REFERENCES `carrier` (`carrierName`)
        ON DELETE NO ACTION
        ON UPDATE NO ACTION,
  CONSTRAINT `portOfCall`
        FOREIGN KEY (`portOfCall`)
        REFERENCES `airport` (`airportCode`)
        ON DELETE NO ACTION
        ON UPDATE NO ACTION,
  CONSTRAINT `portOfDestination`
        FOREIGN KEY (`portOfDestination`)
        REFERENCES `airport` (`airportCode`)
        ON DELETE NO ACTION
        ON UPDATE NO ACTION);


CREATE TABLE `ticket` (
  `bookingNo` INT AUTO_INCREMENT NOT NULL,
  `customerId` INT NOT NULL,
  `flightNo` INT NOT NULL,
  `passportNo` VARCHAR(10) NOT NULL,
  `customerPrice` DECIMAL(6,2) NOT NULL,
  `seatNo` VARCHAR(2) NOT NULL,
```

```
  `bookingDate` DATE NOT NULL,
  PRIMARY KEY (`bookingNo`),
  UNIQUE INDEX `bookingNo_UNIQUE` (`bookingNo` ASC),
  INDEX `customerId_idx` (`customerId` ASC),
  INDEX `flightNo_idx` (`flightNo` ASC),
  CONSTRAINT `customerId`
        FOREIGN KEY (`customerId`)
        REFERENCES `customer` (`customerId`)
        ON DELETE NO ACTION
        ON UPDATE NO ACTION,
  CONSTRAINT `flightNo`
        FOREIGN KEY (`flightNo`)
        REFERENCES `flight` (`flightNo`)
        ON DELETE NO ACTION
        ON UPDATE NO ACTION);


INSERT INTO carrier VALUES('RYANAIR');
INSERT INTO carrier VALUES('FLYBE');
INSERT INTO carrier VALUES('AER LINGUS');
INSERT INTO carrier VALUES('AIR FRANCE');
INSERT INTO carrier VALUES('BRITISH AIRWAYS');


 INSERT INTO customer VALUES(DEFAULT,
'root','Administrator','Root','root@domain.com');
INSERT INTO customer VALUES(DEFAULT, 'hunter2','John','Doe','johndoe@email.com');
```

```
INSERT INTO customer VALUES(DEFAULT,
'strongpassword','Joe','Bloggs','joebloggs@gmail.com');

INSERT INTO customer VALUES(DEFAULT,
'password123','Jane','Doe','janedoe@gmail.com');

INSERT INTO customer VALUES(DEFAULT,
'qwerty','James','Murphy','jamesmurphy@email.com');

INSERT INTO customer VALUES(DEFAULT,
'abc123','Megan','Brown','meganbrown@email.com');


INSERT INTO airport VALUES('ORK', 'CORK AIRPORT', 'CORK');

INSERT INTO airport VALUES('LHR', 'HEATHROW AIRPORT', 'LONDON');

INSERT INTO airport VALUES('CWL', 'CARDIFF AIRPORT', 'CARDIFF');

INSERT INTO airport VALUES('MAD', 'MADRID-BARAJAS ADOLFO SUÁREZ AIRPORT',
'MADRID');

INSERT INTO airport VALUES('CDG', 'PARIS-CHARLES DE GAULLE', 'FRANCE');


INSERT INTO flight VALUES(DEFAULT, 'RYANAIR', 'ORK', 'LHR', '10:00', '10:30',
'2020-01-01', 18);

INSERT INTO flight VALUES(DEFAULT, 'FLYBE', 'ORK', 'CWL', '11:00', '11:30',
'2020-02-02', 36);

INSERT INTO flight VALUES(DEFAULT, 'AER LINGUS', 'CWL', 'LHR', '12:00', '12:30',
'2020-03-03', 36);

INSERT INTO flight VALUES(DEFAULT, 'AIR FRANCE', 'ORK', 'CDG', '13:00', '13:30',
'2020-04-04', 42);

INSERT INTO flight VALUES(DEFAULT, 'BRITISH AIRWAYS', 'ORK', 'MAD', '14:00', '14:30',
'2020-05-05', 110);


INSERT INTO ticket VALUES(1, 1, 1, 'A12345', 18, 'A1', '2019-12-06');

INSERT INTO ticket VALUES(2, 2, 2, 'B12345', 36, 'B1', '2020-01-07');

INSERT INTO ticket VALUES(3, 3, 3, 'C12345', 36, 'C1', '2020-02-08');
```

INSERT INTO ticket VALUES(4, 4, 4, 'D12345', 42, 'D1', '2020-03-09');

INSERT INTO ticket VALUES(5, 5, 5, 'E12345', 110, 'E1', '2020-04-10');

Select Statements

SELECT * FROM airport

airportCode	name	city
CDG	PARIS-CHARLES DE GAULLE	FRANCE
CWL	CARDIFF AIRPORT	CARDIFF
LHR	HEATHROW AIRPORT	LONDON
MAD	MADRID-BARAJAS ADOLFO SUÁREZ AIRPORT	MADRID
ORK	CORK AIRPORT	CORK

SELECT * FROM carrier

carrierName
AER LINGUS
AIR FRANCE
BRITISH AIRWAYS
FLYBE
RYANAIR

SELECT * FROM customer

customerId	password	firstName	lastName	email
1	root	Administrator	Root	root@domain.com
2	hunter2	John	Doe	johndoe@email.com
3	strongpassword	Joe	Bloggs	joebloggs@gmail.com
4	password123	Jane	Doe	janedoe@gmail.com
5	qwerty	James	Murphy	jamesmurphy@email.com
6	abc123	Megan	Brown	meganbrown@email.com

SELECT * FROM flight

flightNo	carrierName	portOfCall	portOfDestination	arrivalTime	departureTime	travelDate	basePrice
1	RYANAIR	ORK	LHR	10:00:00	10:30:00	2020-01-01	18.00
2	FLYBE	ORK	CWL	11:00:00	11:30:00	2020-02-02	36.00
3	AER LINGUS	CWL	LHR	12:00:00	12:30:00	2020-03-03	36.00
4	AIR FRANCE	ORK	CDG	13:00:00	13:30:00	2020-04-04	42.00
5	BRITISH AIRWAYS	ORK	MAD	14:00:00	14:30:00	2020-05-05	110.00

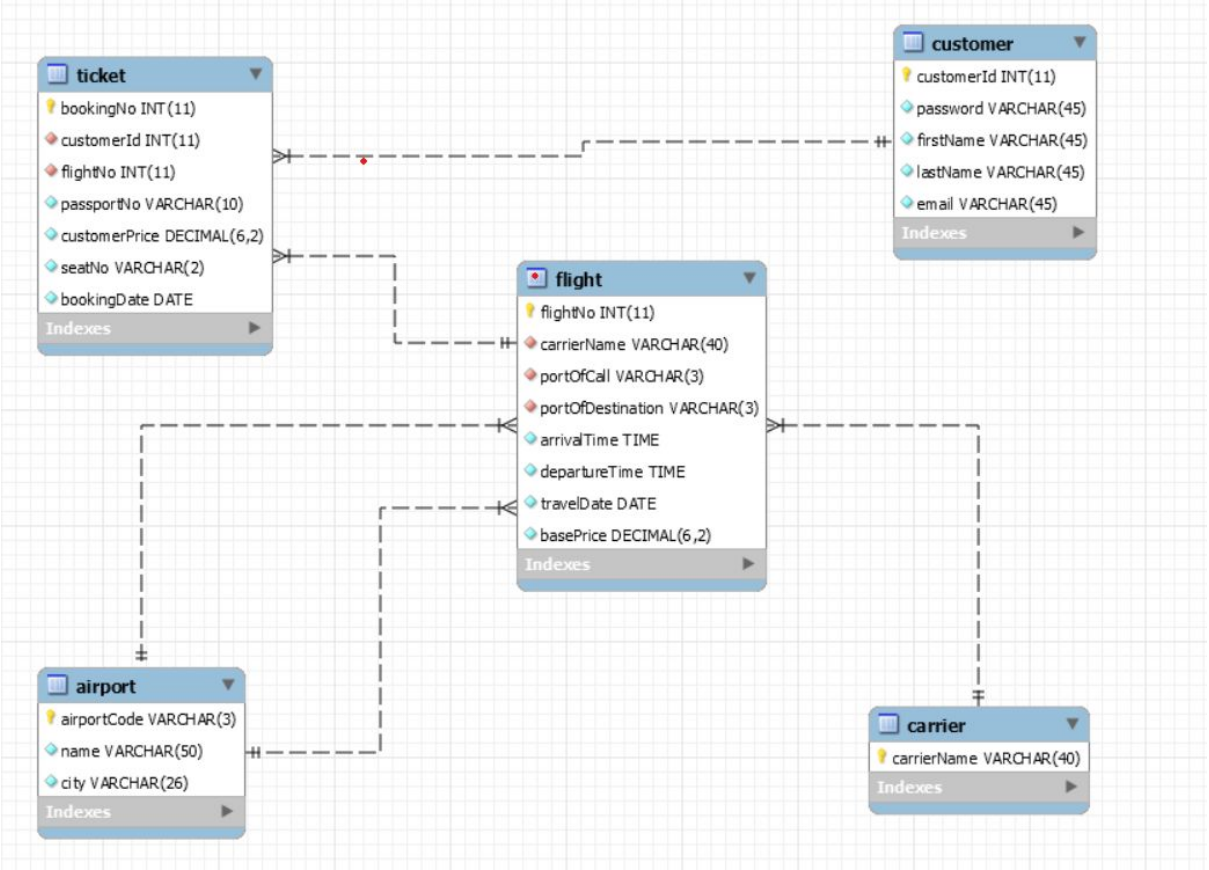
SELECT * FROM ticket

bookingNo	customerId	flightNo	passportNo	customerPrice	seatNo	bookingDate
1	1	1	A12345	18.00	A1	2019-12-06
2	2	2	B12345	36.00	B1	2020-01-07
3	3	3	C12345	36.00	C1	2020-02-08
4	4	4	D12345	42.00	D1	2020-03-09
5	5	5	E12345	110.00	E1	2020-04-10

Technical issues Dealt With

- There were some problems in connecting to the database. It was fixed by downloading the latest driver.
- Hardcoding functions for each table was too time consuming (eg. deleteCustomer(), deleteFlight(), deleteTicket()) was too tedious, especially when accounting for multiple WHERE queries. Switching to ArrayLists helped greatly decrease the workload.
- Had problems in displaying columns in the database. Iterating through the ResultSet object and reading some websites for help and implementing it helped solve the problem.
- Returning data from ResultSet was a problem as it could not be done using a while loop to check if something exists or not as the pointer keeps moving to the next location. Implementing a separate ifExists() function solved this problem.
- Couldn’t compare dates and times using the Date and Time class. Changing the class to LocalDate and LocalTime solved the problem.
- Some logical errors were encountered. Debugging these solved the problem.

Database Design



Five tables were made in the final design of the database.

Some details were repeated in the initial designs (price was included in multiple tables, as was the carrierName). These unnecessary duplications were deleted. There were also some fields that were not required. The airport table had a field for the number of runways, which was later removed as it was not very relevant.

Some tables included in the initial design were also removed. For example, a table having the physical details of the flight (eg. serialNo , fuel, wingspan) was removed as it was not relevant to the application. Another table containing hangar details where flights were stored was also removed.

The carrier table was separated as each carrier can have multiple flights. This makes the flights depend on the name of the carrier. Instead of including it as a composite key, it is put in a separate table as each carrier name should be unique and cannot have a partial dependency on the flight number.

The final version of the table reduces duplication and dependencies, and places the primary key as each table's unique value, making the other fields dependent on it. This also adheres to the rules of normalization.

Java Code Snippets

Insert

```
public void insert(Connection con, String table, ArrayList<Object> myList)
    {
        try
        {
        Statement insertStmt = con.createStatement();

                String query = "INSERT into " + table + " values (";

                for (int i = 0; i < myList.size(); i++)
                {
                        if (i == myList.size() - 1)
                        {
                                query += myList.get(i);
                        }

                        else
                        {
                        query += myList.get(i) + ", ";
                        }


                }
                query += ")";
               int res = insertStmt.executeUpdate(query);
               System.out.println("The Number or records inserted is " +res);
       insertStmt.close();
        }

        catch(Exception io)
        {
                System.out.println("Error: " + io);
        };
    }
```

Update

```
public void update(Connection con, String table, String columnName, Object columnValue,
String primaryColumn, Object primaryValue)
    {
        try
        {
                if(columnValue instanceof String)
                {
                        columnValue = "\'" + columnValue + "\'";
                }

                if(primaryValue instanceof String)
                {
                        primaryValue = "\'" + primaryValue + "\'";
                }

                Statement updateStmt = con.createStatement();
                String updateSQL = "UPDATE " + table + " set " + columnName + " = " +
columnValue + " WHERE " + primaryColumn + " = " + primaryValue;

                int res = updateStmt.executeUpdate(updateSQL);
                System.out.println("The number of records updated is " + res);
                updateStmt.close();
        }

        catch(Exception io)
        {
                System.out.println("Error: " + io);
        }
    }
```

Remove

```
public void delete(Connection con, String table, String row, Object rowValue)
    {
        try
        {
                        if(rowValue instanceof String)
                        {
                                rowValue = "\'" + rowValue + "\'";
                        }

                        Statement deleteStmt = con.createStatement();
         String deleteSQL = " Delete from " + table + " where " + row + " = " + rowValue;
```

```
        int res = deleteStmt.executeUpdate(deleteSQL);
        System.out.println("The Number or records deleted is " +res);
        deleteStmt.close();
        }
        catch(Exception io)
        {
        System.out.println("error"+io);
        };
    }
```

Basic Query

```
public void queryStatement(Connection con, String table, ArrayList<Object> myList)
  {
        try {
                        String columnValues = "";

                        for (int i = 0; i < myList.size(); i++)
                                {
                                        if (i == myList.size() - 1)
                                        {
                                                columnValues += myList.get(i);
                                        }

                                        else
                                        {
                                                columnValues += myList.get(i) + ", ";
                                        }

                                }

                Statement stmt = con.createStatement();
                        String query = "SELECT "  + columnValues + " FROM " + table;
                        ResultSet rs = stmt.executeQuery(query);

                        //https://coderwall.com/p/609ppa/printing-the-result-of-resultset
                        ResultSetMetaData rsmd = rs.getMetaData();
                        int columnsNumber = rsmd.getColumnCount();
                        while (rs.next()) {
                           for (int i = 1; i <= columnsNumber; i++) {
                              if (i > 1) System.out.print(",  ");
                              String columnValue = rs.getString(i);
                              System.out.print(rsmd.getColumnName(i) + ": " + columnValue);
                           }
                           System.out.println("");
```

```
                    }
        }

        catch(Exception io)
        {
                System.out.println("Error: " + io);
        };
}
```

Application Design

The application was designed using the console as a basis for interactivity. It focuses on two functions: A customer-centric function and and administrative-centric function.

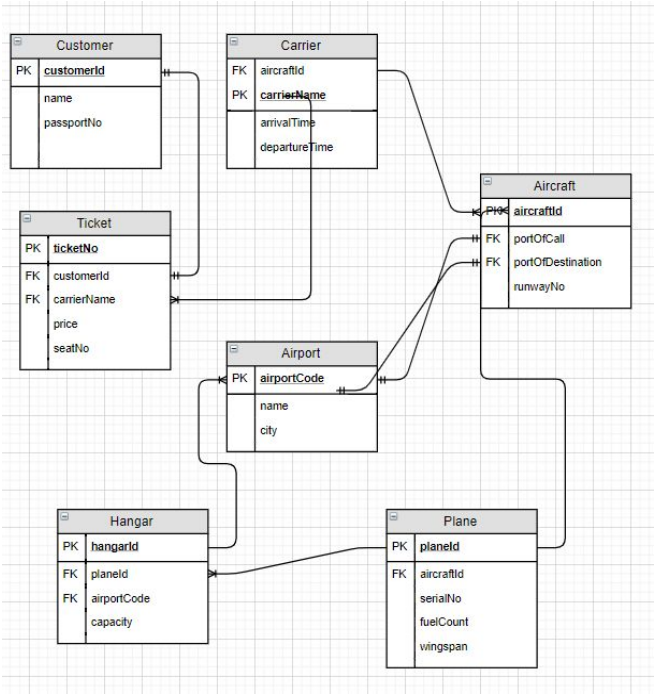
The application asks for the customer's details when asked to book or cancel a ticket, and asks for the administrator's details when performing administrative functions.

A number of checks and validations are also implemented into the application to prevent logical inconsistencies. For example, the booking date cannot be after the travel date, and the departure airport cannot be the same as the arrival airport.

The application's functions are built on using ArrayLists and iterating through them to insert the values into the database. This is much more efficient than having one CREATE-READ-UPDATE-DELETE function for each individual table.

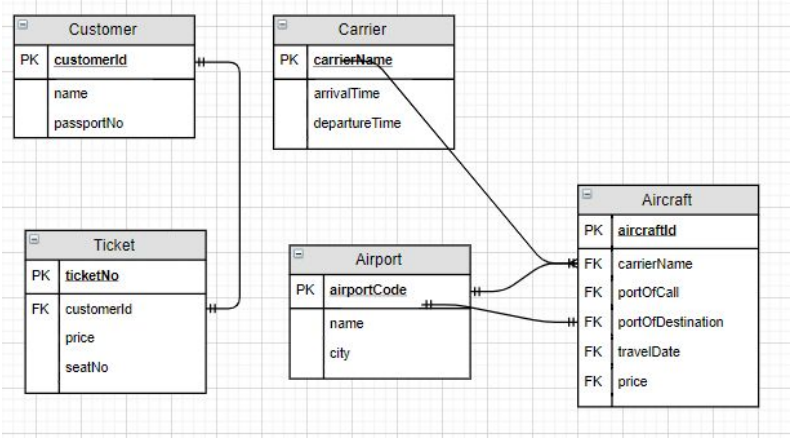
Log

- **19th October**: Made a draft ER diagram with 7 tables.



-

- **21st October**: Got feedback on draft ER diagram. Removed plane table, removed unnecessary fields in carrier table and airport table.

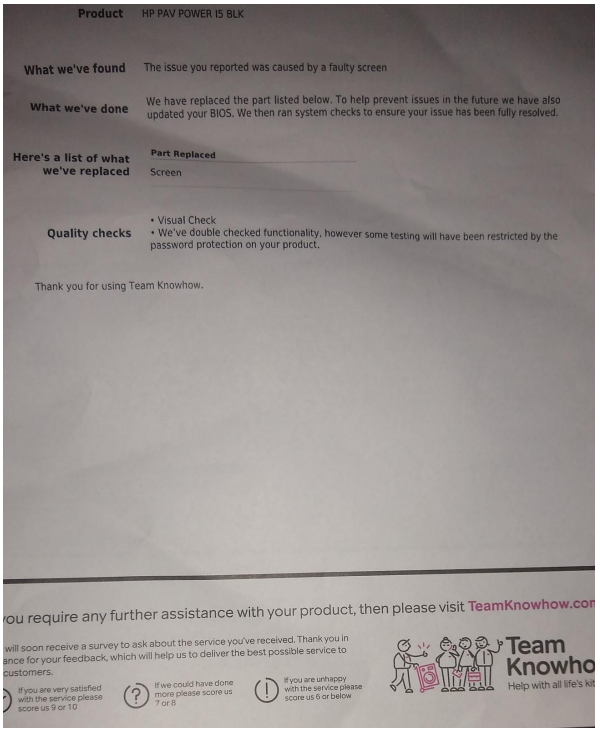


-

- **28th October**: Changed the diagram again after feedback (removed unnecessary fields). Generated source code from Visual Paradigm from the ER diagram. Created a new database connection and input CREATE statements. Problem connecting to the database in the lab, but fixed it later by using the latest connector.
- **29th October**: Populated the database with 5 inputs for each table.
- **30th October**: Added inserts for all tables with different functions for each (eg. insertCustomer(), insertBooking())
- **4th November**: Laptop broke and sent in for repairs. Have to restart Java part on another laptop as I didn't back it up.
- **7th November**: Borrowed spare laptop for use. Plan to implement CRUD functions using ArrayLists instead of hardcoding them for each table, saving time and reducing lines of code. Implemented the insert class using ArrayLists. Made a sample console menu.
- **14th November**: Changed the database source code to use AutoIncrement as it was too tedious to generate a random number for each table with a unique id. Changed

console menu after feedback in lecture. Implemented customer creation and login functions

- **17th November**: Added function to check if something exists in the database. Implemented password checking, made it possible to get values from the database and store them in a variable. Implemented function to create a customer. Implemented a function to generate a random seat number.
- **18th November**: Implemented pricing function based on the booking date. Implemented single and multiple conditional-query functions (with WHERE). Implemented ticket cancellation function.
- **20th November**: Added validation to passwords, flight time, flight dates. Added comments describing each function. Deleted old functions and classes that were unused.
- **23nd November**: Received repaired laptop. Transferred project files from the other laptop and checked to see if it works.



Product	HP PAV POWER I5 BLK
What we've found	The issue you reported was caused by a faulty screen
What we've done	We have replaced the part listed below. To help prevent issues in the future we have also updated your BIOS. We then ran system checks to ensure your issue has been fully resolved.
Here's a list of what we've replaced	**Part Replaced** Screen
Quality checks	• Visual Check • We've double checked functionality, however some testing will have been restricted by the password protection on your product.

Thank you for using Team Knowhow.

ou require any further assistance with your product, then please visit TeamKnowhow.co

will soon receive a survey to ask about the service you've received. Thank you in ance for your feedback, which will help us to deliver the best possible service to customers.

If you are very satisfied with the service please score us 9 or 10

If we could have done more please score us 7 or 8

If you are unhappy with the service please score us 6 or below

Team Knowho

Help with all life's kit

- **24th November**: Implemented other sections of report and formatted them. Tested all functions to check if they work.

Conclusions/Review

This project was very useful in learning about database design. I learnt how to create an ER diagram with the relevant connectors. I learnt how to optimize the diagram and remove unnecessary tables/fields and duplicate entries.

I learnt about useful database functions such as auto increment and about the sqlite java connector. This was useful in connecting the java front-end to the database.

This project was also useful in terms of documentation and logging activities that help keep track of progress each week, providing encouragement to complete the assignment.

References/Guides

1. https://coderwall.com/p/609ppa/printing-the-result-of-resultset
2. http://www.mysqltutorial.org/mysql-jdbc-update
3. https://docs.oracle.com/javase/tutorial/jdbc/basics/retrieving.html
4. https://www.tutorialspoint.com/jdbc/jdbc-where-clause.htm
5. https://www.callicoder.com/java-arraylist/
6. https://stackoverflow.com/questions/26711383/getting-one-value-from-sql-select-statement-in-java/26711421#26711421
7. https://docs.oracle.com/javase/8/docs/api/java/time/LocalDate.html
8. https://docs.oracle.com/javase/8/docs/api/java/time/LocalTime.html

Statement

I hereby certify that this material which I now submit for assessment, is entirely my own work and has not been taken from the work of others, save and to the extent that such work has been cited and acknowledged within the text of my work

_____

Signed