

**Tutor-marked Assignment 3**

**TCC 233/05 Database Management Systems**

**Name : KHOR CHUAN PING**

**Student ID : 041150052**

**Class Code : DMS1-5C**

**Contact No : 017-5050747**

**Email : kcp4\_pg@student.wou.edu.my**

**Submission Date: 29th March, 2023 (2400 hrs)**

**Tutor : VIMALA A/P DORAISAMY**

**Course Coordinator: CHANDARASAGERAN S/O NATARAJAN**

**Regional Centre : Penang**

**No. of Pages : 15**

**(Including this & Declaration Form)**

****

**PLAGIARISM DECLARATION FORM (T-DF)**

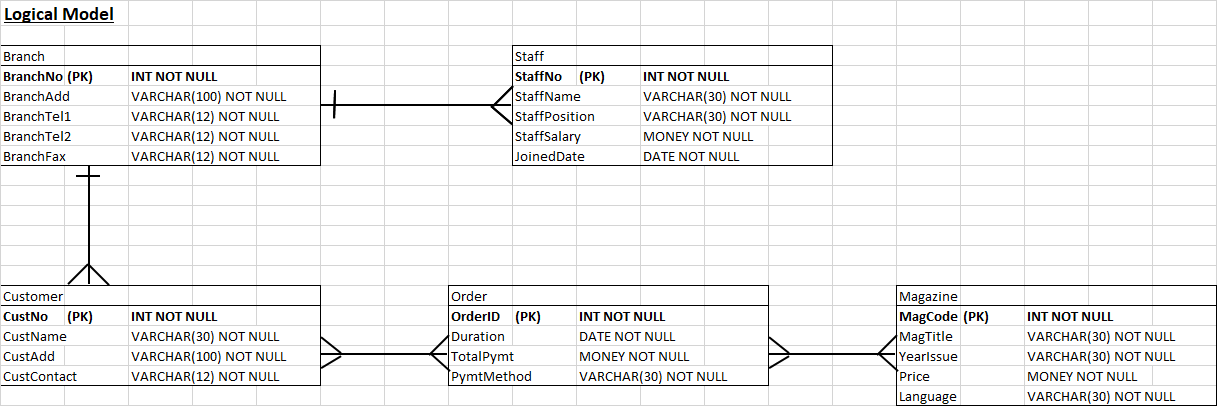
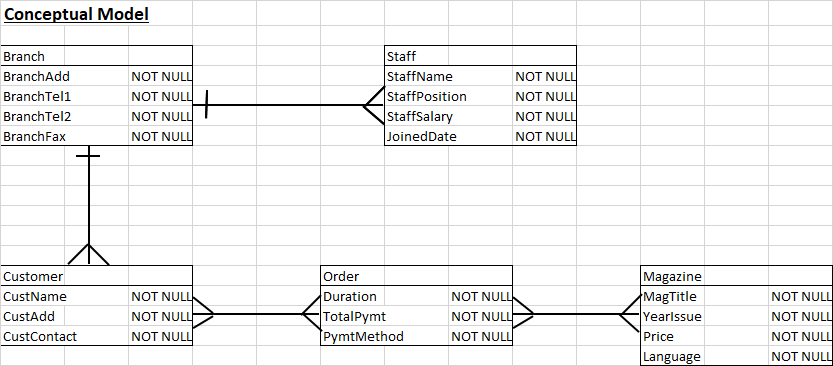
|  |
| --- |
| **Instructions**  **Please complete and attach this Plagiarism Declaration Form to each Assignment that you submit into the Online Assignment Submission (OAS) system for marking.** |
| I declare that the attached work is entirely my own (or when submitted to meet the requirements of an approved group assignment is the work of the group), except where materials cited, quoted or paraphrased are acknowledged in the text. I also declare that this work / assignment has not been submitted for assessment in any other course or university without due acknowledgement.  I understand that plagiarism, collusion, and copying are grave and serious offences.  I understand that disciplinary action (which may include deduction of marks in the Assignment) will be taken against me if I am found to be an offender of Assignment plagiarism.  **Full name and IC No: KHOR CHUAN PING (880512-35-5015) Date: 23rd March 2023** |

|  |  |
| --- | --- |
| **Assignment (Asgmt) Declaration Form** | |
| Semester/Year | 1/2023 |
| Student’s Name | KHOR CHUAN PING |
| Student’s ID No: | 041150052 |
| Course Code | TCC233/05 |
| Course Title | Database Management Systems |
| Class Code | DMS1-5C |
| Assignment No: | 3 |
| No. of pages of this Assignment *(including this page)* | 15 |
| Tutor | VIMALA A/P DORAISAMY |
| Course Coordinator | CHANDARASAGERAN S/O NATARAJAN |

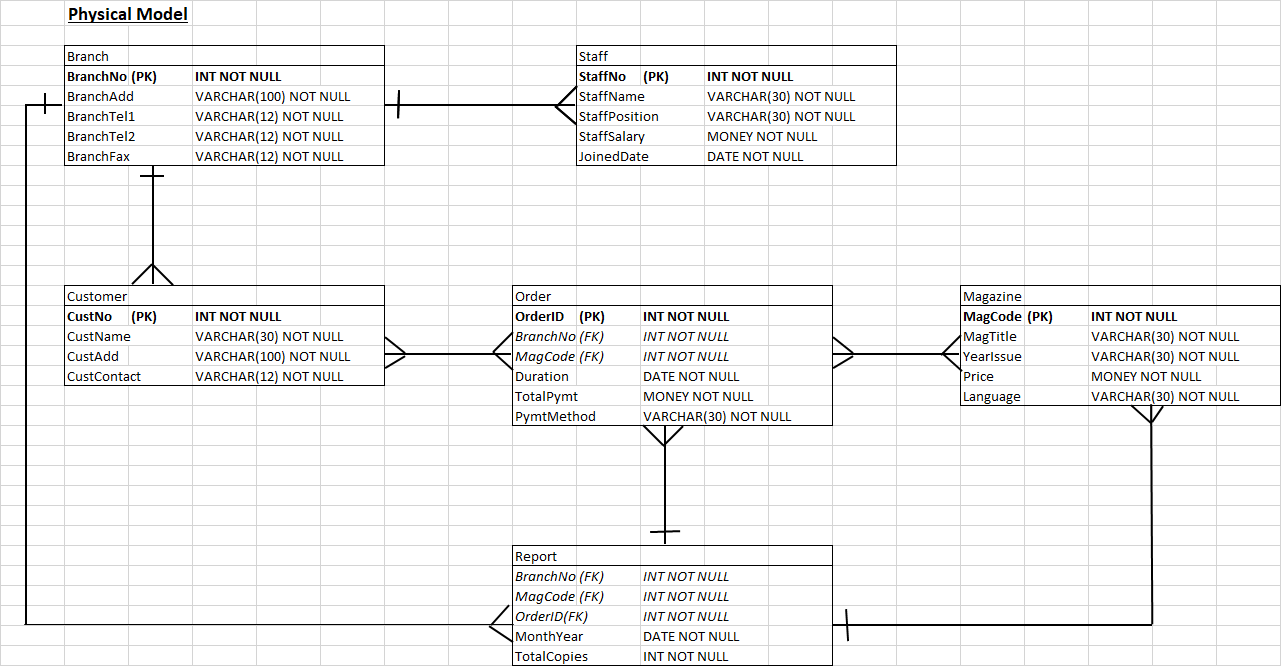
***T-DF* *Assignment Declaration Form (1/2020 version #003)***

**Question 1a**



****







**Question 1b**

1. Data Manipulation Language (DML), DML commands are used to manipulate data in a database, in the scenario, examples as below:

The INSERT & SELECT command:

INSERT INTO Branch (BranchTel2)

VALUES (012-5447813)

SELECT BranchTel1, BranchTel2 FROM Branch;

This allows to add new info into current table tuple



1. Data Definition Language (DDL), DDL commands are used to define the structure of a database and its objects, to add table from the scenario, examples as below:

The CREATE command:

CREATE TABLE Staff

(

StaffNo INT NOT NULL PRIMARY KEY,



StaffName VARCHAR(30) NOT NULL,

StaffPosition VARCHAR(30) NOT NULL,

StaffSalary MONEY NOT NULL,

JoinedDate DATE NOT NULL

);

This used to create new table for MAGAZINE DISTRIBUTOR COMPANY

1. Data Control Language (DCL), DCL commands are used to control access and permissions on a database and its objects, examples as below:

The GRANT command:

GRANT SELECT ON Staff TO Report;



This command grants Staff to access Report table for the information

**Question 1c**

The approach taken for the handling of data in early file-based systems, no matter in digital or physical form of filling system by the MAGAZINE DISTRIBUTOR COMPANY, they doing so much bookkeeping and filling work to keep data up to date.

The company’s staff have to store all details such as:

‘Branch’ stores each branch details such as:

Branch number, address, contact, fax

‘Staff File’ stores staff personal details such as:  
 Staff number, name, position, salary, joined date

‘Customer’ stores customer personal details such as:

Customer number, name, address, contact

‘Order’ stores customer’s purchase information such as:

OrderID, branch number, magazine code, subscription duration, total payment, payment method

‘Magazine’ stores each issued magazine information such as:

Magazine code, magazine title, year and month of issue, price, language

‘Report’ is a monthly generated order statistic record that contains information such as:

Branch number, magazine code, orderID, month and year, total of copies sold group by branch

* The ‘Branch’ file contains staff information in specific branch, it can calculate 1 branch have how many staff in the file.



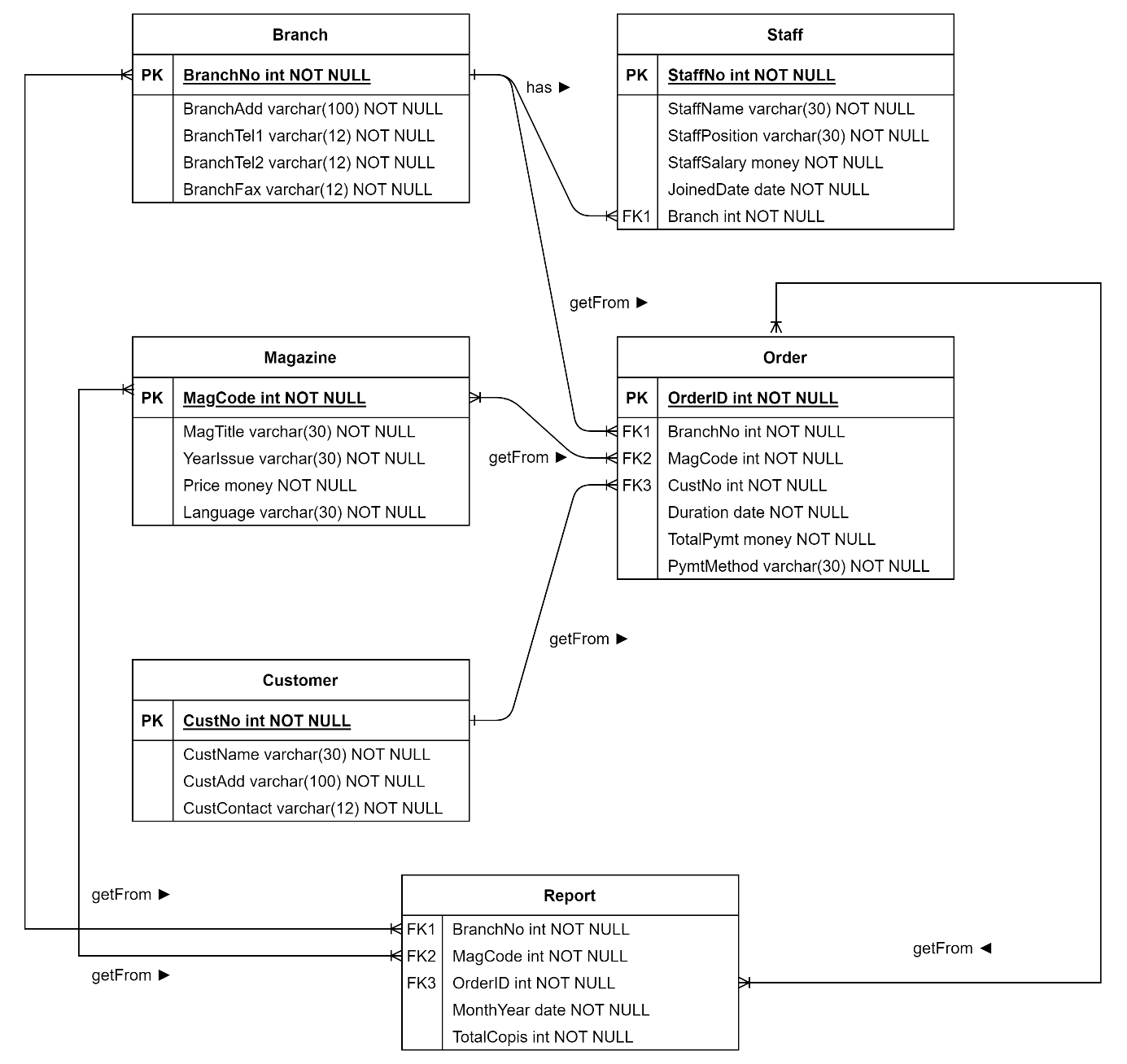
* + Branch1 = {staff1, staff2, staff3}
  + Branch2 = {staff4, staff5, staff6}
* The ‘Order’ relates branch of purchased, which magazine ordered, and personal details that customer ordered.
  + Order1 = {Branch1, Customer1, Magazine4, May2022 – Apr2023, 12 x RM8 TOTAL 96, Credit Card}
  + Order2 = {Branch2, Customer3, Magazine2, Jan2022 – June2022, 6 x RM 10 TOTAL 60, Cash}
* The ‘Report’ file calculates the statistic of monthly each magazine ordered sum amount
  + ReportJan2022 = {Branch1, Magazine1, Feb2022, 50 Sold},

{Branch1, Magazine2, Mar2022, 45 Sold},

{Branch2, Magazine1, Feb2022, 30 Sold}



**Question 2**





****

**Question 3a**

1. A base table is a database object that stores data in rows and columns, each table have primary key (pk) and some tables have foreign key (fk) for referencing to another table. The graphic above shows it have three base tables which are:

* Student Table, contains id as primary key, sName, address, state, contactNo, sex, email as balance attributes without key or constraints.



* Module Table, contains id as primary key, mName, creditHour, preReqID, lecID as balance attributes. preReqID and lecID may have another base table after normalization form.
* Marks Table, this base tables don’t have primary key attributes but with two foreign key that referencing data from Student table and Module table, there is only one normal attribute to record marks.



A view table can be based on one or more base tables, with the SQL command (DML, DDL, DCL) usages, to generate customised view table by calling base table data. Relational Algebra and Boolean Algebra knowledges are required to build command to generate a view table.



* The view table consists of
  + Student table (id as Student ID, sName as Name)
  + Module table (mName as Module)
  + Marks table (moduleID reference Module.id, marks as Marks)



1. The purpose of having this view table is to generate a convenient and efficient way to read the table.

* For example, referencing Marks table, it only shows studentID moduleID and marks. It is unreadable for lecturer and student while they want to announce the result.
* To simplify the table, it has to be generated into a view table that convenient for lecturer and student. So, the view table should contain:



* + Student ID (this reference from StudentTable.ID, select sName as Name to show in view table)
  + Module (need to reference from moduleID first, and then select Module to show, EXCEPT moduleID to show)
  + Marks (only SELECT marks for view table, referencing Marks table with composite key (studentID, moduleID)

1. There are some restrictions when updating the view table:

* It can’t modify data in views table based on union queries
  + For example, studentID from Marks table is referenced from Student table id, if a view table generated by Marks table, it have union between two tables, are unable to modify the data
* It can’t update aggregate functions or DISTINCT clause columns as it is calculated from original data
  + For example, the view table SUM how many modules each student took, it will show data invalid
* It only can update one table in the view if the view table contains joins between two or more tables.
  + For example, a view table consists of Student table and Module table, we can update the address and state in Student table but we can’t update address from Student table, and marks from Marks table at the same time



* If the data from base tables violates any constraints, rules, or referential integrity defined, it not able to update
  + For example, we can’t update studentID and moduleID from Marks table with foreign key constraint
* It can’t update the data that conflicts WITH CHECK OPTION clause, but it can use NO CHECK for possible views.
  + For example, the Marks base table has a mark attribute while the view table use HighMarks where the marks is greater than 80, it referenced from Marks table attribute marks. It can’t update the marks as it will fail because it tries to update the marks attribute in the view to a value that does not satisfy the view definition



**Question 3b**

1. σ from = “Kuala Lumpur” (Flight)



1. π seatNo, customerNo (σ price > 2500 (Seat ⋈ Booking))
2. count(σ customerName = “Nichole” (Customer ⋈ Booking))



**Question 4**



* 1. Toys(Code) is primary key and also is candidate key



* 1. Toys table is first normal form because attributes don’t have repeat data in a tuple, such as:
  + Toy(Code, ToyName), toyname can be same but code are unique



* + Toy(Producer, ProducesAdd) are unique so probably won’t have two address in a tuple



* + Toy(TotalOrder, TotalInStock) value is based from each unique code



* + Toy(DateProduce), each toy has only one production date, so it won’t happen multiple date in same tuple
  + Toy(ToyName, Category) paired to determine the category, single toy that break to multiple category will cause stock value conflict



* + Toy(SellingPrice, Cost), it may have multiple selling price but cost is fixed



* 1. Valid Functional Dependencies
  + FD1: Code → ToyName



* + FD2: Producer → ProducesAdd



* + FD3: TotalOrder → TotalInStock
  + FD4: SellingPrice → Cost



* + FD5: Code, ToyName → DateProduce, Category



* + FD6: Code → ToyName, Designer, Producer, ProducesAdd, TotalOrder, TotalInStock, DateProduce, Category, SellingPrice



* 1. Insertion anomaly: Code dependent on production date, if a toy is created but half way done, there is no production date for the data entry, it cannot insert first because Code needs to check if there are multiple ToyName with different production date, a duplicate of a toy is given a new Code.



Delete anomaly: If a toy is discontinued, Designer resigned, Producer closed, deleting these related records can result in the unintentional loss of data. If a designer resigned, admin delete the designer’s name, this will impact to toy list and caused all toys under this designer will also be deleted.



* 1. The relation contains few possible transitive dependencies, for example:



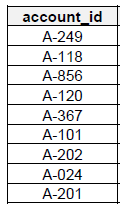
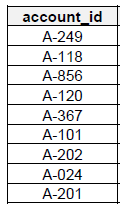
* + ToyName → Code → DateProduce
  + Code → Producer → ProducesAdd



**Question 5a**

Heap Files stores records in the same order they are inserted.

Raw Data Heap File Block

1

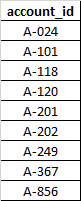
2

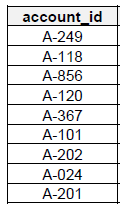
3



Sequential File stores and sorts the records on ordering field.

Raw Data Sequential File Block





1

2



3

Hash File stores record to the block address generated by a hash function

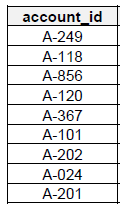
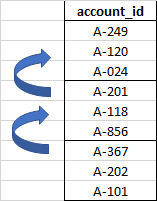
For example, account\_id = 249

Hash Function id mod 3

249/3 = 0

Record 249 stores in Block 0

Raw Data Hash File Block

0

1

2

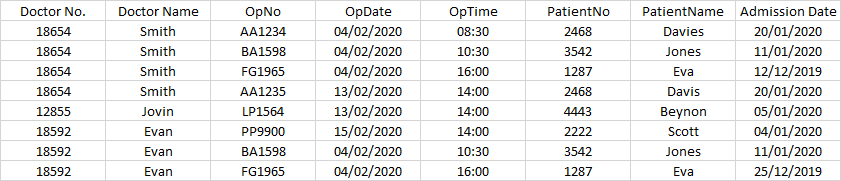


A-201 overflow from 0 to 1  
 A-367, A-202 overflow from 1 to 2

**Question 5b**

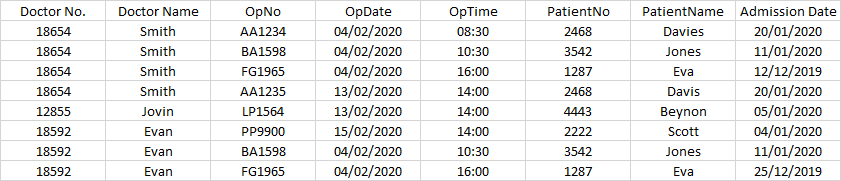
**UNF**

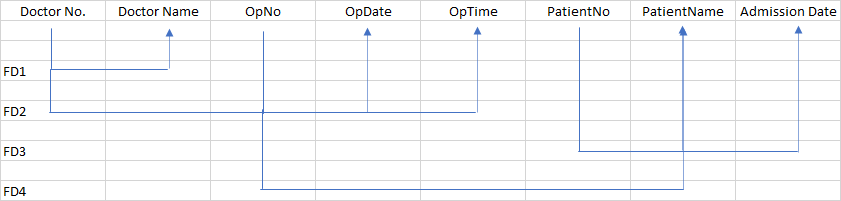
**Operating Schedule**

****



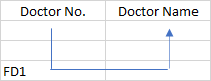
**1NF**

**Operating Schedule**

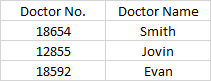
****

**2NF**

FD1: DoctorNo. → DoctorName

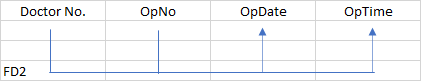
  
**Doctor List**



****

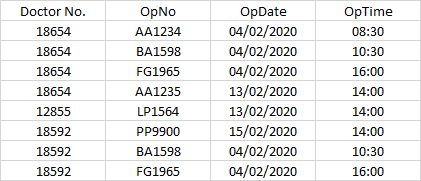


FD2: DoctorNo., OpNo → OpDate, OpTime

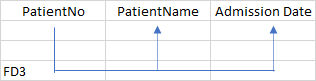




**Operation List**

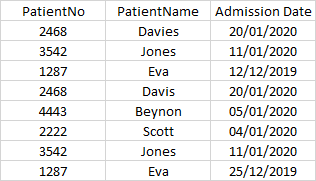
****

FD3: PatientNo → PatientName, Admission Date

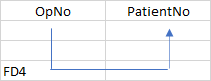




**Patient List**

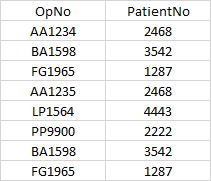
****

FD4: OpNo → PatientNo

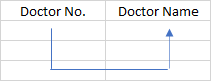




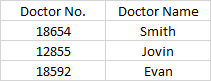
**Operation List**

****

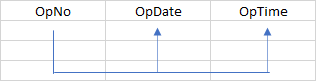
**3NF**

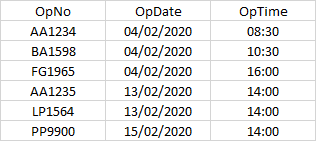
****

**Doctor List**

****



**  
Operation Schedule**

****

