Pusat Pengajian Pengkomputeran

School of Computing

Universiti Utara Malaysia

STID3013 DATABASE SYSTEM AND INFORMATION RETRIEVAL
Group F

Semester I Session 2022/2023 (A221)

DATABASE SYSTEM PROJECT: MUSICAL INSTRUMENT STORE

SUBMITTED BY:

GROUP 4

GROUP MEMBERS:

277209	NG PEI NYUK
278171	CHONG CHING WEI
279620	LAU JIA MIN

Lecturer Name:

Dr. Syahida Binti Hassan

Submission Date:
24 January 2023

TABLE OF CONTENTS

1.0 BUSINESS RULE

Mrs. Michelle is the owner of a musical instrument stall in Changlun, Malaysia. She used bookkeeping to drop down every information (which include customer information, staff information, invoices, etc) by handwriting. As the information is getting bigger and the bookkeeping is no longer efficient, therefore she needs a database management system to store the existing and new information.

Firstly, the system will save the general information for the staff, which includes staffID, staffName, staffAddress, workingHour, and salary. For salary, the charge will be $8 per hour. The salary will be given to all the staff every week. The database system should be able to save customer information, which include customerID, customerName and customerAddress.

One staff can have one or many customers while the customer did purchase in the stall. While one customer will only have one staff that will be serving he/she.

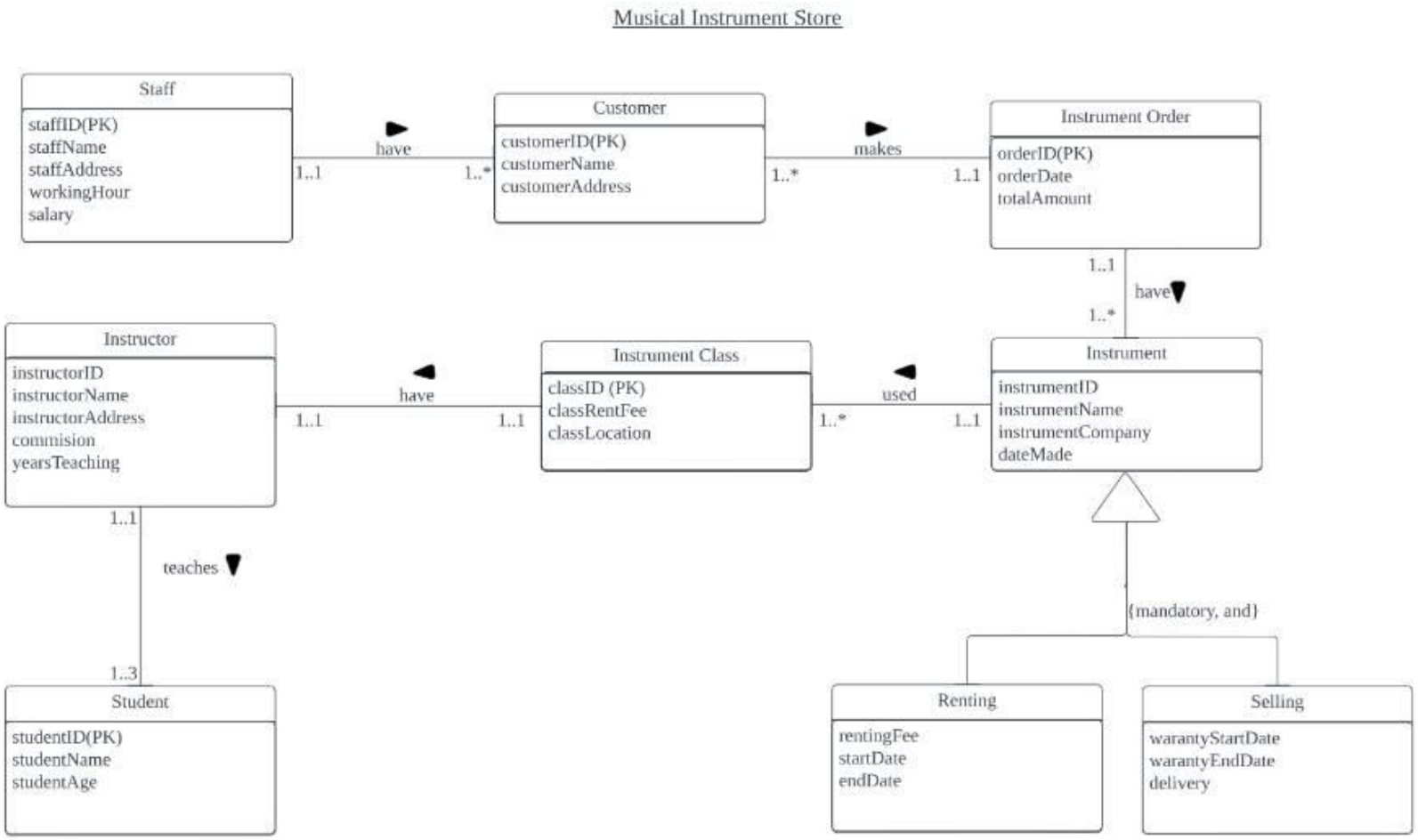
An invoice will be given by the staff when the customers did the purchase on the instrument. The invoice will state orderID, date, and total price for the instrument. If the customer wishes to have delivery services, extra charges will be charged in the invoice. Every purchase for 1 instrument will be given 1 year of warranty.

Every instrument will be recorded by their instrumentID that is unique, following with the instrumentName, instrumentCompany, and the date that it was made. The instrument can be sold or rented by customers.

As the students are required to bring their own instrument, therefore the instruments will be used in instrument class by the instructor that recruits from outsiders. Instructors are required to have at least 2-years working experience. 1 class will have at maximum 3 person students. The instrument class should have classID, class renting fee, and class location.

The database system should record the general information for the instructor, which is instructorID, instructorName, instructorAddress, and the class he/she teaches. General Information for students, which is studentID, studentName, and studentAge should be saved inside the system. Every month, students will pay the fee to the instructor, and the instructor will pay the renting fee to the staff. The remaining fee is his/hers commission.

2.0 Entity Relationships Diagram (ERD)



3.0 Relational Model (RM)

Staff (staffID, staffName, staffAddress, workingHour, salary)
Primary Key: staffID

Customer (customerID, customerName, customerAddress, staffID, orderID)
Primary Key: customerID
Foreign Key: staffID references Staff(staffID)
Foreign Key: orderID references InstrumentOrder(orderID)

InstrumentOrder (orderID, orderDate, totalAmount)
Primary Key: orderID

Student (studentID, studentName, studentAge, instructorID)
Primary Key: studentID
Foreign Key: instructorID references Instructor(instructorID)

Instructor (instructorID, instructorName, instructorAddress, commission,
instrumentID,classID,classLocation,classRentFee, yearsTeaching)
Primary Key: instructorID
Foreign Key: instrumentID references Instrument(instrumentID)

Instrument(instrumentID, instrumentName, instrumentCompany, dateMade, rentingFee,
startDate, endDate, warrantyStartDate, warrantyEndDate, delivery, orderID)
Primary Key: instrumentID
Foreign Key: orderID references InstrumentOrder(orderID)

4.0 SQL QUERIES

4.1 CREATE TABLES (Queries For All Table)

```
CREATE TABLE Staff (
staffID int NOT NULL,
staffName VARCHAR(40) NOT NULL,
staffAddress VARCHAR(40) NOT NULL,
workingHour int NOT NULL,
salary DECIMAL(9,2) NOT NULL,
CONSTRAINT Staff_PK
                PRIMARY KEY (staffID)
);
```

This command is used for creating a table named Staff. This table will record all required data that is related to the staff that are working in the musical stall. The important data such as ID, which is unique and it will differentiate the data, full name of the staff, current address, working hour and the salary. For the salary, salary will be given within a week, and it will be multiplied with $8 per hour.

```
CREATE TABLE InstrumentOrder (
orderID int NOT NULL,
orderDate DATE NOT NULL,
totalAmount DECIMAL(9,2) NOT NULL,
CONSTRAINT InstrumentOrder_PK
                PRIMARY KEY (orderID),
);
```

This command is made to record the invoice when the customer makes an order about the instrument. Every invoice will be unique by creating an order ID that will not be duplicated. At the same time, the date and the total amount for the invoice will be recorded too.

```
CREATE TABLE Customer (
customerID int NOT NULL,
staffID int NOT NULL,
customerName VARCHAR(40) NOT NULL,
customerAddress VARCHAR(40) NOT NULL,
orderID int NOT NULL,
CONSTRAINT Customer_PK
                PRIMARY KEY (customerID),
CONSTRAINT Staff_Customer_PK
                FOREIGN KEY (staffID)
                REFERENCES Staff(staffID),
CONSTRAINT InstrumentOrder_Customer_PK
                FOREIGN KEY (orderID)
                REFERENCES InstrumentOrder(orderID),
);
```

This command is used to create a table which will be able to save customers' data. When every customer has been purchased in invoice, the database will record a unique ID for each customer. Other information such as customer name and address will be recorded too. The staff that serves them will be recorded inside this table by using staff ID. In the database, the order that has been made by customers will also be recorded in the table by using order ID.

```
CREATE TABLE Instrument(
instrumentID int NOT NULL,
orderID int NOT NULL,
instrumentName VARCHAR(40) NOT NULL,
instrumentCompany VARCHAR(40) NOT NULL,
dateMade DATE NOT NULL,
rentingFee DECIMAL (3,2) NOT NULL,
startDate DATE NOT NULL,
endDate DATE NOT NULL,
warrantyStartDate DATE NOT NULL,
warrantyEndDate DATE NOT NULL,
delivery DECIMAL (5,2) NOT NULL,
CONSTRAINT Instrument_PK
                PRIMARY KEY (instrumentID),
CONSTRAINT Instrument_Order_PK
                FOREIGN KEY (orderID)
                REFERENCES InstrumentOrder(orderID),
);
```

This command is used to create a table which will be able to save the information about the instruments. In this table, it will contain a unique key and will not be duplicated in each tuple, and this key will be named as instrumentID. Table Customer will be linked with table Order as 1 order will have 1 or more than 1 instrument that will be purchased by the customers. Therefore, orderID, which is the primary key for table Order, will be the foreign key for this table, which will show the relationship in both tables. The information related with the instrument will be recorded too, such as instrument name, instrument company and the date made for the instrument. For the service, the store provided 2 services, which is renting and selling service. For renting service, the fee for the renting, the start and the end date will be recorded too. Meanwhile for selling service, the warranty start and end date, and delivery fee will be recorded. As the service is mandatory and will be chosen, therefore all of the information will be recorded in the table Instrument.

```
CREATE TABLE Instructor (
instructorID int NOT NULL,
instrumentID int NOT NULL,
instructorName VARCHAR(40) NOT NULL,
instructorAddress VARCHAR(40) NOT NULL,
commission int NOT NULL,
classID int NOT NULL,
classLocation VARCHAR(30) NOT NULL,
classRentFee VARCHAR(3) NOT NULL,
yearsTeaching int NOT NULL,
CONSTRAINT Instructor_PK
                PRIMARY KEY (instructorID),
CONSTRAINT Instructor_Instrument_PK
                FOREIGN KEY (instrumentID)
                REFERENCES Instrument(instrumentID),
);
```

The database also will record the information about the Instructor that will teach the instrument. Therefore, besides the unique key, namely instructorID has been recorded, instrumentID will also be recorded inside the table. Therefore, there is a relationship between 2 tables. Instructor information, such as name, address, years teaching and commission will be recorded too. In the table, classID, classLocation and classRentFee will be included too.

```
CREATE TABLE Student (
studentID int NOT NULL,
instructorID int NOT NULL,
studentName VARCHAR(40) NOT NULL,
studentAge int NOT NULL,
CONSTRAINT Student_PK
                PRIMARY KEY (studentID),
CONSTRAINT Student_Instructor_PK
                FOREIGN KEY (instructorID)
                REFERENCES Instructor(instructorID),
);
```

The database also will store the student information in a table named as Student. This table will record all the data related to the student who registered the instrument class, such as studentID, as a unique key, student's name and student's age. The table also will store the instructor ID, which refers to the instructor that will teach in the instrument class.

4.2 ALTER TABLE

```
CREATE TABLE Instrument(
instrumentID int NOT NULL,
orderID int NOT NULL,
instrumentName VARCHAR(40) NOT NULL,
instrumentCompany VARCHAR(40) NOT NULL,
dateMade DATE NOT NULL,
rentingFee VARCHAR(2) NOT NULL,
startDate DATE NOT NULL,
endDate DATE NOT NULL,
warrantyStartDate DATE NOT NULL,
warrantyEndDate DATE NOT NULL,
delivery VARCHAR(4) NOT NULL,
CONSTRAINT Instrument_PK
                PRIMARY KEY (instrumentID),
CONSTRAINT Instrument_Order_PK
        FOREIGN KEY (orderID)
        REFERENCES InstrumentOrder(orderID),

);
```

Table Instrument will be updated by using the ALTER function in SQL. The ALTER TABLE statement is used to add, delete, or modify columns in an existing table. For this table, we have modified the attributes, which is rentingFee and delivery fee, from DECIMAL data type to VARCHAR data type. This is because the data will not be changed in future, therefore the VARCHAR data type is suitable in this case.

Below is the comparison for the **before** ALTER and **after** ALTER. The command also had been written below the before ALTER section. For example, we wrote a command using ALTER TABLE and ALTER COLUMN to modify the datatype of the existing column which is 'rentingFee' from VARCHAR(2) to VARCHAR(3) as shown below.

BEFORE:

OUTPUT:

	instrumentID	orderID	instrumentName	instrumentCompany	dateMade	rentingFee	startDate	endDate	warrantyStartDate	warrantyEndDate	delivery
1	1	511	Guitar	Ibanez	2008-05-24	90	2019-10-25	2019-12-25	2019-10-25	2020-10-25	100
2	2	512	Keyboard	Yamaha	2005-09-29	80	2021-04-01	2021-06-01	2021-04-01	2022-04-01	100
3	3	513	Violin	Stentor	1999-01-21	90	2018-07-13	2018-09-13	2018-07-13	2019-07-13	100
4	4	514	Drum	Tama	2017-12-31	90	2022-06-14	2022-09-14	2022-06-14	2023-06-24	240
5	5	515	Piano	Yamaha	1995-04-18	90	2020-03-25	2020-05-25	2020-03-25	2021-03-25	780
6	6	516	Piano	Yamaha	1995-04-18	90	2020-03-25	2020-05-25	2020-03-25	2021-03-25	780

COMMAND:

```
ALTER TABLE Instrument
ALTER COLUMN rentingFee VARCHAR (3);
```

AFTER:

OUTPUT :

instrumentID	orderID	instrumentName	instrumentCompany	dateMade	rentingFee	startDate	endDate	warrantyStartDate	warrantyEndDate	delivery
1	511	Guitar	Ibanez	2008-05-24	90	2019-10-25	2019-12-25	2019-10-25	2020-10-25	100
2	512	Keyboard	Yamaha	2005-09-29	80	2021-04-01	2021-06-01	2021-04-01	2022-04-01	100
3	513	Violin	Stentor	1999-01-21	90	2018-07-13	2018-09-13	2018-07-13	2019-07-13	100
4	514	Drum	Tama	2017-12-31	90	2022-06-14	2022-09-14	2022-06-14	2023-06-24	240
5	515	Piano	Yamaha	1995-04-18	90	2020-03-25	2020-05-25	2020-03-25	2021-03-25	780
6	516	Piano	Yamaha	1995-04-18	90	2020-03-25	2020-05-25	2020-03-25	2021-03-25	780
7	516	Flute	ArmStrong	2021-03-22	100	2021-09-23	2021-09-23	2021-07-23	2021-09-23	180

4.3 INSERT ALL DATA

```
-- Table Staff
INSERT INTO Staff VALUES ('110', 'Lily Tan', 'Changlun, Kedah', '8', (8*8*7));
INSERT INTO Staff VALUES ('111', 'Ali', 'Alor Setar, Kedah', '10', (10*8*7));
INSERT INTO Staff VALUES ('112', 'Eric', 'Sik, Kedah', '5', (5*8*7));
INSERT INTO Staff VALUES ('113', 'Jason', 'Kampung Baru, Kuala Lumpur', '10', (10*8*7));
INSERT INTO Staff VALUES ('114', 'Amuthu', 'Kulai, Johor', '6', (6*8*7));
INSERT INTO Staff VALUES ('115', 'Mizan', 'Kota Kinabalu, Sabah', '12', (12*8*7));
```

This INSERT INTO function allows the owner of the database to input the data that is needed by the Staff table into the database. Following the data type and the structure of the table Staff that had been created earlier, therefore the data inserted by follow the sequence which is staffID (in integer), staffName (in VARCHAR), staffAddress (in VARCHAR), workingHours (int) and the derived attribute, which is salary, that will be calculated while the DBMS inserting the data.

```
--Table Customer
INSERT INTO Customer VALUES ('801','115', 'Katie', 'Changlun, Kedah','511');
INSERT INTO Customer VALUES ('802','111', 'Abu', 'Pulau Langkawi, Kedah','512');
INSERT INTO Customer VALUES ('803','112', 'Tan', 'Baling, Kedah','513');
INSERT INTO Customer VALUES ('804','113', 'Anusha', 'Kulim, Kedah','514');
INSERT INTO Customer VALUES ('805','114', 'Yusof', 'Bukit Mertajam, Penang','515');
INSERT INTO Customer VALUES ('806','115', 'Patrisya', 'Gurun, Kedah','516');
INSERT INTO Customer VALUES ('808','113', 'Chee', 'Seberang Perai, Penang','517');
```

This query will let the owner insert the data into the Customer table. As the table Customer that had been created earlier by using the INSERT INTO command, therefore the data inserted by follow the sequence which is customerID (in integer), staffID (in integer) that had been retrieved from table Staff, customerName (in VARCHAR), customerAddress (in VARCHAR), orderID(int) that had been retrieved from table Order.

```
--INSTRUMENT ORDER
INSERT INTO InstrumentOrder VALUES ('511', '2022-06-17', '400');
INSERT INTO InstrumentOrder VALUES ('512', '2022-04-28', '350');
INSERT INTO InstrumentOrder VALUES ('513', '2021-10-23', '500');
INSERT INTO InstrumentOrder VALUES ('514', '2022-02-15', '450');
INSERT INTO InstrumentOrder VALUES ('515', '2021-12-08', '700');
INSERT INTO InstrumentOrder VALUES ('516', '2020-02-14', '960');
INSERT INTO InstrumentOrder VALUES ('517', '2020-02-14', '1000');
```

This INSERT INTO function will insert the data into the InstrumentOrder table. As all the invoice's data can't be saved in the database, therefore the owner decided to insert the important data into the database. The data will be inserted by following the sequence that had been created earlier which are orderID(in integer), orderDate (in date type), and totalAmount (in decimal type with length 9, and 2 decimal).

```
--INSTRUMENT
INSERT INTO Instrument VALUES ('01',
'511','Guitar','Ibanez','2008-05-24','90','2019-10-25','2019-12-25','2019-10-25','2020-10-25','100'
);
INSERT INTO Instrument VALUES ('02', '512','Keyboard','Yamaha','2005-09-29' ,'80'
,'2021-04-01' ,'2021-06-01', '2021-04-01', '2022-04-01' , '100');
INSERT INTO Instrument VALUES ('03', '513','Violin','Stentor','1999-01-21' ,'90' ,'2018-07-13'
,'2018-09-13', '2018-07-13', '2019-07-13' , '100');
INSERT INTO Instrument VALUES ('04', '514','Drum','Tama','2017-12-31' ,'90' ,'2022-06-14',
'2022-09-14', '2022-06-14', '2023-06-24' , '240');
INSERT INTO Instrument VALUES ('05', '515','Piano','Yamaha','1995-04-18' ,'90' ,'2020-03-25'
,'2020-05-25', '2020-03-25','2021-03-25' , '780');
INSERT INTO Instrument VALUES ('06', '516','Trumpet','GETZAN', '2022-12-12' ,'90'
,'2023-01-01' ,'2025-01-01','2022-12-25', '2023-01-25' , '90');
INSERT INTO Instrument VALUES ('07', '516','Flute','ArmStrong','2021-03-22' ,'100'
,'2021-09-23' ,'2023-09-23', '2021-07-23', '2021-09-23' , '180');
INSERT INTO Instrument VALUES ('08', '517','Bass','Fender','2020-06-18' ,'100' ,'2020-07-18'
,'2023-07-18', '2021-09-01', '2021-10-01' , '80');
```

This INSERT INTO function will let the data to be inserted into the Instrument table. All the data required will be inserted inside the table. As mentioned in the create table section, the table had been combined as the relationship of the participation of both sides is mandatory participation, therefore the subtable, which is renting instruments and selling instruments, will be combined with the parent table, which is Instrument table. The data will be inserted by following the sequence that had been created earlier which are instrumentID (in integer), orderID(in int) that will be retrieved from the table Order, instrumentName (in VARCHAR), instrumentCompany (in VARCHAR), dateMade (in date), rentingFee (in integer), startDate (in date), endDate (in date), warrantyStartDate(in date), warrantyEndDate(in date), and delivery (in VARCHAR).

```
--Instructor
INSERT INTO Instructor VALUES ('301','01','Ying', 'Cheras, Selangor', '400','901','CL01','350',
'3');
INSERT INTO Instructor VALUES ('302','02', 'Edwin', 'Georgetown, Pulau Pinang',
'400','902','CL02','350', '9');
INSERT INTO Instructor VALUES ('303','03', 'Bryan', 'Ipoh, Perak', '400','903','CL03','350', '5');
INSERT INTO Instructor VALUES ('304','04', 'Michelle', 'Batu Pahat, Johor',
'400','904','CL04','350', '5');
INSERT INTO Instructor VALUES ('305','05', 'Aina', 'Changlun, Kedah', '400','905','CL05','350',
'2');
INSERT INTO Instructor VALUES ('306','06', 'Laveniya', 'Puchong, Selangor',
'400','906','CL06','350', '2');
```

This INSERT INTO function will let the data to be inserted into the instructor table. All the data required will be inserted inside the table. As the relationship is 1 to 1, therefore the class's data will be inserted in the table instructor. The data will be inserted by following the sequence that

had been created earlier which are instructorID (in integer), instrumentID (in int) that will be retrieved from the table Instrument, instructorName (in VARCHAR), instructorAddress (in VARCHAR), commission (in integer), classID (in integer), classLocation (in VARCHAR), and yearsTeaching (in integer).

```
--Student
INSERT INTO Student  VALUES('601','302', 'Jenny', '15')
INSERT INTO Student  VALUES('602','305', 'Shamimi', '12')
INSERT INTO Student  VALUES('603','301', 'Amran', '17')
INSERT INTO Student  VALUES('604','303', 'Siew Ying', '20')
INSERT INTO Student  VALUES('605','302', 'Mira', '16')
INSERT INTO Student  VALUES('606','304', 'Alfred', '14')
INSERT INTO Student  VALUES('607','301', 'Ning', '13')
INSERT INTO Student  VALUES('608','302', 'Asyraf', '22')
```

This INSERT INTO function will let the data be inserted into the student table. The data will be inserted by following the sequence that had been created earlier which are studentID (in integer), instructorID (in int) that will be retrieved from the table Instructor, studentName (in VARCHAR) and studentAge (in integer).

4.4 UPDATE ANY DATA

COMMAND:

```
UPDATE InstrumentOrder
SET totalAmount = '15000'
WHERE orderID = '516'
```

The data in the table InstrumentOrder will be updated by using the UPDATE function in SQL. The UPDATE statement is used to modify the existing records in a table.

Below is the comparison for the **before** and **after** for UPDATE function. The command also had been written as above. For example, we wrote a command using UPDATE function to modify the data of the existing column which is 'totalAmount' from 23100 to 15000 as shown below.

BEFORE

	orderID	orderDate	totalAmount
1	511	2022-06-17	400.00
2	512	2022-04-28	350.00
3	513	2021-10-23	500.00
4	514	2022-02-15	450.00
5	515	2021-12-08	700.00
6	516	2020-02-14	23100.00
7	517	2020-02-14	1000.00

AFTER

	orderID	orderDate	totalAmount
1	511	2022-06-17	400.00
2	512	2022-04-28	350.00
3	513	2021-10-23	500.00
4	514	2022-02-15	450.00
5	515	2021-12-08	700.00
6	516	2020-02-14	15000.00
7	517	2020-02-14	1000.00

4.5 CREATE FIVE VIEWS

```
CREATE VIEW Customer_Staff113 AS
SELECT * FROM Customer
WHERE staffID IN (
SELECT staffID FROM Staff
WHERE staffID = '113'
)
```

For view 'Customer_Staff113', the query had been created by using the data included in the database. This view is created to retrieve the data which is the customer that had been served by the staff that have the ID '113'. Therefore, the query will select all the columns that the database saved, and get the specific condition, which is staffID equals to '113' that has been recorded in table staff. Below is the output created:

	customerID	staffID	customerName	customerAddress	orderID
1	804	113	Anusha	Kulim, Kedah	514
2	808	113	Chee	Seberang Perai, Penang	517

```
CREATE VIEW Instructor_305 AS
SELECT stud.studentName, inst.instructorName,ins.instrumentName
FROM Student stud, Instructor inst, Instrument ins
WHERE stud.instructorID = inst.instructorID
AND inst.instrumentID = ins.instrumentID
AND inst.instructorID = '305'
```

For view 'Instructor_305', the query had been created by using the data included in the database. This view is created to retrieve the data which is the student name, the instructor name, and the instrument name with the instructorID is equals with '305'. Therefore, the query will select the studentName from table Student, instructorName from table Instructor, and instrumentName from table Instrument. The data will compare with the foreign keys in the tables with primary keys. Therefore, the data will filter the data that fulfill the WHERE and AND clause. Below is the output created:

	studentName	instructorName	instrumentName
1	Shamimi	Aina	Piano

```
CREATE VIEW student_classLoc AS
SELECT stu.studentName, inst.instrumentID, inst.classLocation
FROM Student stu, Instructor inst, Instrument ins
WHERE inst.instructorID = stu.instructorID
AND inst.instrumentID = ins.instrumentID
AND ins.instrumentName = 'guitar'
```

For view 'student_classLoc', the query had been created by using the data included in the database. This view is created to retrieve the student name, the instrument id, and also the classLocation with a condition that the instrument name is equal to 'guitar'. Therefore, the

query will select studentName from table Student, instrumentName from table Instrument, and classLocation from table Instructor. The data will compare with the foreign keys in the tables with primary keys. Therefore, the data will filter the data that fulfill the WHERE and AND clause. Below is the output created:

	studentName	instrumentID	classLocation
1	Amran	1	CL01
2	Ning	1	CL01

```
CREATE VIEW All_Staff_customer AS
SELECT stf.staffName, cus.customerName, cus.orderID
FROM Staff stf, Customer cus
WHERE stf.staffID = cus.staffID
```

For view 'All_Staff_customer', the query had been created by using the data included in the database. This view is created to retrieve the staff name, the customer name, and also the order id. Therefore, the query will select staffName from table Staff, customerName and orderID from table Customer. The data will compare with the foreign keys in the tables with primary keys. Therefore, the data will filter the data that fulfill the WHERE clause. Below is the output created:

	staffName	customerName	orderID
1	Mizan	Katie	511
2	Ali	Abu	512
3	Eric	Tan	513
4	Jason	Anusha	514
5	Amuthu	Yusof	515
6	Mizan	Patrisya	516
7	Jason	Chee	517

```
CREATE VIEW instrument_detail AS
SELECT InstrumentOrder.orderID,InstrumentOrder.orderDate,Instrument.instrumentName
FROM InstrumentOrder, Instrument
WHERE InstrumentOrder.orderID = Instrument.orderID
```

For view ‘instrument_detail’, the query had been created by using the data included in the database. This view is created to retrieve the order id, the order date, and also the instrument name. Therefore, the query will select orderID,orderDate from table InstrumentOrder and instrumentName from table Instrument. The data will compare with the foreign keys in the tables with primary keys. Therefore, the data will filter the data that fulfill the WHERE clause. Below is the output created:

	orderID	orderDate	instrumentName
1	511	2022-06-17	Guitar
2	512	2022-04-28	Keyboard
3	513	2021-10-23	Violin
4	514	2022-02-15	Drum
5	515	2021-12-08	Piano
6	516	2020-02-14	Trumpet
7	516	2020-02-14	Flute
8	517	2020-02-14	Bass

4.6 DROP ANY VIEW

BEFORE:

Before starting the query, the view named as student_classLoc had been created earlier. The view involves 3 tables, which are table student, table instrument and table instructor with the data needed, which is studentName from table student, instrumentID from table Instrument and classLocation from table Instructor. Below is the command and the output for this view:

COMMAND:

```
CREATE VIEW student_classLoc AS
SELECT stu.studentName, inst.instrumentID, inst.classLocation
FROM Student stu, Instructor inst, Instrument ins
WHERE inst.instructorID = stu.instructorID
AND inst.instrumentID = ins.instrumentID
AND ins.instrumentName = 'guitar'
```

	studentName	instrumentID	classLocation
1	Amran	1	CL01
2	Ning	1	CL01

The view will be dropped by using the DROP function in SQL. The DROP statement is used to drop, which means deleting existing records in a table.

AFTER:

COMMAND:

```
DROP VIEW student_classLoc
```

Below is the **after** for the DROP function. The command also had been written as above. Query has been written using the DROP function to delete the data of the existing view which is student_classLoc. The type will be followed behind the function student, which is VIEW and followed with the view name, which is student_classLoc. Below is the output for the command:

OUTPUT:

```
Messages
Msg 208, Level 16, State 1, Line 32
Invalid object name 'student_classLoc'.

Completion time: 2023-01-18T00:16:38.5551696+08:00
```

4.7 CREATE SUBQUERIES

```
SELECT instructorID,instructorName,instructorAddress,commission FROM Instructor
WHERE instrumentID IN (
SELECT instrumentID FROM Instrument
WHERE orderID = '516'
)
```

This query is written to select the item needed and display it in the table at the output section. As the data needed is instructor information, which is instructorID, instructorName, instructorAddress and commission that is orderID equal to '516'. Therefore, the query will try to find the needed data by using foreign key, which is the primary key for the table Instrument and get the data. Below is the output for the query:

	instructorID	instructorName	instructorAddress	commission
1	306	Laveniya	Puchong, Selangor	400

```
SELECT staffID,staffName FROM Staff
WHERE staffID IN (
SELECT staffID FROM Customer
WHERE customerID = '803'
)
```

This query is written to select the item needed and display it in the table at the output section. As the data needed is staff information, which is staffID and staffName where customerID is equal to '803'. Therefore, the query will try to find the needed data by using the primary key, which is the foreign key for the table Customer and get the data. Below is the output for the query:

	staffID	staffName
1	112	Eric

```
SELECT * FROM Instructor
WHERE instructorID IN (
SELECT instructorID FROM Student
WHERE studentID = '601'
)
```

This query is written to select the item needed and display it in the table at the output section. As the data needed is the details of Instructor where studentID is equal to '601'. Therefore, the query will try to find the needed data by using the primary key, which is the foreign key for the table Student and get the data. Below is the output for the query:

	instructorID	instrumentID	instructorName	instructorAddress	commission	classID	classLocation	classRentFee	yearsTeaching
1	302	2	Edwin	Georgetown, Pulau Pinang	400	902	CL02	350	9

```
SELECT * FROM STAFF
WHERE staffID IN (SELECT staffID FROM Customer
WHERE orderID IN (
SELECT orderID FROM InstrumentOrder
WHERE orderDate = '2022-06-17'
)
)
```

This query is written to select the item needed and display it in the table at the output section. As the data needed is the details of staff where the orderDate is equal to '2022-06-17'. Therefore, the query will try to find the needed data by using the primary key, staffID which is the foreign key for the table Customer in order to get the primary key, orderID which is the foreign key for InstrumentOrder to get the data. Below is the output for the query:

	staffID	staffName	staffAddress	workingHour	salary
1	115	Mizan	Kota Kinabalu, Sabah	12	672.00

```
SELECT instrumentID,instrumentName,instrumentCompany FROM Instrument
WHERE instrumentID IN (
SELECT instrumentID FROM Instructor
WHERE instructorID IN (
SELECT instructorID FROM Student
WHERE studentName = 'Jenny'
)
)
```

This query is written to select the item needed and display it in the table at the output section. As the data needed is instrumentID, instrumentName and instrumentCompany where studentName is equal to 'Jenny'. Therefore, the query will try to find the needed data by using the primary key, instrumentID which is the foreign key for the table Instructor in order to get the primary key, instructorID which is the foreign key for Student to get the data. Below is the output for the query:

	instrumentID	instrumentName	instrumentCompany
1	2	Keyboard	Yamaha

4.8 CREATE MULTI-TABLE QUERIES

```
SELECT ins.instrumentName,ins.instrumentCompany, inst.instructorName, inst.yearsTeaching
FROM Instrument ins, Instructor inst, student stu
WHERE inst.instrumentID = ins.instrumentID
AND inst.instructorID = stu.instructorID
AND stu.studentName = 'Mira'
```

This query is written to select the item needed and display it in the table at the output section. This query contains 3 tables, which are table Instrument, table Instructor and table Student. As the data needed is information of instrument such as instrumentName instrumentCompany, and information of instructor such as instructorName and yearsTeaching, with a condition where the studentName is equal to ‘Mira’. Therefore, the query will try to find the needed data by using foreign key, which is the primary key for the table Instrument and get the data. Below is the output for the query:

	instrumentName	instrumentCompany	instructorName	yearsTeaching
1	Keyboard	Yamaha	Edwin	9

```
SELECT cus.customerName, cus.staffID, ord.totalAmount, stf.staffName
FROM Customer cus, InstrumentOrder ord, staff stf
WHERE cus.orderID = ord.orderID
AND cus.staffID = stf.staffID
AND ord.orderDate = '2020-02-14'
```

This query is written to select the item needed and display it in the table at the output section. This query contains 3 tables, which are table Customer, table InstrumentOrder and table Staff. As the data needed is information of customer such as customerName and staffID, information of InstrumentOrder which is the totalAmount, and information of staff which is staffName, with a condition where the orderDate is equal to '2020-02-14'. Therefore, the query will try to find the needed data by using foreign key, which is the primary key for the table Instrument and get the data. Below is the output for the query:

	customerName	staffID	totalAmount	staffName
1	Patrisya	115	960.00	Mizan
2	Chee	113	1000.00	Jason

```
SELECT cus.customerName, ord.orderDate
FROM Customer cus, InstrumentOrder ord, Instrument ins
WHERE cus.orderID = ord.orderID
AND ins.orderID = ord.orderID
AND ins.instrumentName = 'piano'
```

This query is written to select the item needed and display it in the table at the output section. This query contains 3 tables, which are table Customer, table InstrumentOrder and table Instrument. As the data needed is information of customer which is customerName, information of InstrumentOrder which is the orderDate, with a condition where the instrumentName is equal to 'piano'. Therefore, the query will try to find the needed data by

using foreign key, which is the primary key for the table Instrument and get the data. Below is the output for the query:

	customerName	orderDate
1	Yusof	2021-12-08

```
SELECT stu.studentName, stu.studentAge, inst.classLocation
FROM Student stu, Instructor inst, Instrument ins
WHERE inst.instructorID = stu.instructorID
AND classID = '901'
AND ins.instrumentName = 'guitar'
```

This query is written to select the item needed and display it in the table at the output section. This query contains 3 tables, which are table Student, table Instructor and table Instrument. As the data needed is information of student such as studentName and studentAge, information of instructor which is the classLocation, with a condition where the classID is equal to '901' and instrumentName is equal to 'guitar'. Therefore, the query will try to find the needed data by using foreign key, which is the primary key for the table Instrument and get the data. Below is the output for the query:

	studentName	studentAge	classLocation
1	Amran	17	CL01
2	Ning	13	CL01

```
SELECT ins.instrumentName, ins.instrumentCompany, ins.dateMade, ord.orderDate,
inst.classID, inst.instructorName
FROM Instrument ins, Student stu, Instructor inst,InstrumentOrder ord
WHERE ins.instrumentID = inst.instrumentID
AND inst.instructorID = stu.instructorID
AND ins.orderID = ord.orderID
AND ins.instrumentName = 'violin'
```

This query is written to select the item needed and display it in the table at the output section. This query contains 3 tables, which are table Instrument, table Student, table Instructor and table InstrumentOrder. As the data needed is information of instument such as instrumentName, instrumentCompany and dateMade,information of InstrumentOrder which is orderDate, and also the information of instructor such as the classID and instructorName, with a condition where instrumentName is equal to 'violin'. Therefore, the query will try to find the needed data by using foreign key, which is the primary key for the table Instrument and get the data. Below is the output for the query:

	instrumentName	instrumentCompany	dateMade	orderDate	classID	instructorName
1	Violin	Stentor	1999-01-21	2021-10-23	903	Bryan

4.9 CREATE QUERIES INCLUDES AGGREGATE FUNCTION

```
SELECT COUNT (orderID) AS TotalOrder
FROM InstrumentOrder
```

This query is written to count the order that has been made. The query will count the total row for the table order when the row is not empty and has a primary key. This data will be rename as TotalOrder. Below is the output for the query:

	TotalOrder
1	7

```
SELECT staffName, staffID,
(SELECT MIN (salary)
FROM Staff) AS Salary
FROM Staff
WHERE salary = (SELECT MIN (salary)
FROM Staff)
```

This query is written to get the minimum salary that the staff get. The query will make a table as an output, the table should display the staffName and the staffID at the same time. The query will get the staff who has the most minimum salary. The column will be renamed as salary. Therefore, the query will try and get the data for the staff. Below is the output for the query:

	staffName	staffID	Salary
1	Eric	112	280.00

```
SELECT orderID, orderDate,
(SELECT AVG (totalAmount)
FROM InstrumentOrder) AS AverageAmount
FROM InstrumentOrder
WHERE orderID = '516'
```

This query is written to get the average for the order. The query will make a table as an output, the table should display the orderID and the orderDate at the same time. The query will get the average for the order and rename it as AverageAmount. Therefore, the query will try and get the data for the staff. Below is the output for the query:

	orderID	orderDate	AverageAmount
1	516	2020-02-14	622.857142

```
SELECT instrumentID, instrumentName,
(SELECT MAX (rentingFee)
FROM Instrument) AS rentingFee
FROM Instrument
WHERE rentingFee = (SELECT MAX (rentingFee)
FROM Instrument)
```

This query is written to get the maximum rentingFee for the Instrument. The query will make a table as an output, the table should display the instrumentID and the instrumentName at the same time. The query will get the staff who has the most maximum rentingFee. The column will be renamed as rentingFee. Therefore, the query will try and get the data for the staff. Below is the output for the query:

	instrumentID	instrumentName	rentingFee
1	1	Guitar	90
2	3	Violin	90
3	4	Drum	90
4	5	Piano	90
5	6	Trumpet	90

```
SELECT SUM (commission) AS Total_Commission
FROM Instructor
```

This query is written to count and get the sum for the commission. The query will get the sum of the value that is inside the commission column. This data will be renamed as Total_Commission. Below is the output for the query:

	Total_Commission
1	2400