Additional SQL Commands

USERS

DB_ADMIN

```
CREATE USER 'db_admin'@'localhost' IDENTIFIED BY 'password';
GRANT ALL PRIVILEGES ON * . * TO 'db_admin'@'localhost';
FLUSH PRIVILEGES;
```

Web_Developer

```
CREATE USER 'web_dev'@'localhost' IDENTIFIED BY 'password';
GRANT SELECT, INSERT, UPDATE, DELETE, ALTER, CREATE, DROP
ON a5.* TO 'web_dev'@'localhost';
FLUSH PRIVILEGES;
```

End_User

```
CREATE USER 'end_user'@'localhost' IDENTIFIED BY 'password';
GRANT SELECT, INSERT, UPDATE, DELETE
ON a5.* TO 'end_user'@'localhost';
FLUSH PRIVILEGES;
```

Stored Procedure

Nurse

```
DROP PROCEDURE IF EXISTS nurseInsert;
DELIMITER //
Create Procedure nurseInsert
    (IN PersonRole VARCHAR(255), FirstName VARCHAR(255), LastName
VARCHAR(255), Address VARCHAR(255), Birthdate VARCHAR(255),
PhoneNumber VARCHAR(255), EmployeeType VARCHAR(255), CareCenterID
VARCHAR(255), Certificate VARCHAR(255))
Begin
    Insert Into Person ( `PersonRole`, `FirstName`, `LastName`, `Address`,
`Birthday`, `PhoneNumber`)
    Values (PersonRole, FirstName, LastName, Address, Birthdate,
PhoneNumber);

INSERT INTO `Employee` (`EmployeeType`, `CareCenterID`, `PersonID`) VALUES
(EmployeeType, 11000, LAST_INSERT_ID());

INSERT INTO `Nurse` ( `Certificate`, `EmployeeID`)
 VALUES ( Certificate, LAST_INSERT_ID());

End//
DELIMITER ;
```

EXAMPLE CALL:
CALL nurseInsert ("Nurse", "Barb", "Wendys", "555 McDonalds Road", "2009-07-30", "305-928-6654x79", "b", 11000, "Yes");
Technician

```
DROP PROCEDURE IF EXISTS technicianInsert;
DELIMITER //
Create Procedure technicianInsert
    (IN PersonRole VARCHAR(255), FirstName VARCHAR(255), LastName VARCHAR(255), Address VARCHAR(255), Birthdate VARCHAR(255), PhoneNumber VARCHAR(255), EmployeeType VARCHAR(255), CareCenterID VARCHAR(255), Skill VARCHAR(255))
Begin
    Insert Into Person ( `PersonRole`, `FirstName`, `LastName`, `Address`, `Birthday`, `PhoneNumber`)
    Values (PersonRole, FirstName, LastName, Address, Birthdate, PhoneNumber);

INSERT INTO `Employee` (`EmployeeType`, `CareCenterID`, `PersonID`) VALUES (EmployeeType, 11000, LAST_INSERT_ID());

Insert Into Technician ( `EmployeeID`, `Skill`)
    Values (LAST_INSERT_ID(), `Mathematics`);

End//
DELIMITER ;
```

Volunteer

```
DROP PROCEDURE IF EXISTS volunteerInsert;
DELIMITER //
Create Procedure volunteerInsert
    (IN PersonRole VARCHAR(255), FirstName VARCHAR(255), LastName VARCHAR(255), Address VARCHAR(255), Birthdate VARCHAR(255), PhoneNumber VARCHAR(255), EmployeeType VARCHAR(255), CareCenterID VARCHAR(255), Skill VARCHAR(255))
Begin
    Insert Into Person ( `PersonRole`, `FirstName`, `LastName`, `Address`, `Birthday`, `PhoneNumber`)
    Values (PersonRole, FirstName, LastName, Address, Birthdate, PhoneNumber);

INSERT INTO `Employee` (`EmployeeType`, `CareCenterID`, `PersonID`) VALUES (EmployeeType, 11000, LAST_INSERT_ID());

Insert Into Volunteer ( `PersonID`, `Skill`)
    Values (LAST_INSERT_ID(), `Janitor`);

End//
DELIMITER ;
```

Triggers

```
CREATE TRIGGER `after_person_insert` AFTER INSERT ON `person`
FOR EACH ROW BEGIN INSERT INTO person_backup
VALUES(NEW.PersonID, NEW.PersonRole, NEW.FirstName, NEW.LastName,
NEW.Address, NEW.Birthday, NEW.PhoneNumber);
END
```

All Records added to person will also be added to person_backup as well after the trigger is implemented. Requires the person_backup table to exist already with same fields(done in SQL2B).

Views

A Nurse view to see the nurse's information, and most importantly if they have there certificate or not.

```
CREATE VIEW nurseView AS
SELECT FirstName, LastName, PersonRole, Certificate, PhoneNumber,
employee.EmployeeID
FROM Person, Nurse, Employee
WHERE nurse.EmployeeID = employee.EmployeeID AND employee.PersonID =
person.PersonID;
```

A Nurse view to see the head nurse information especially to check if they are certified to work there.

```
CREATE VIEW nurseInChargeView AS
SELECT FirstName, LastName, PersonRole, Certificate, PhoneNumber,
employee.EmployeeID
FROM carecenter, Person, Nurse, Employee
WHERE nurse.EmployeeID = employee.EmployeeID AND employee.PersonID =
person.PersonID AND nurse.NurseID = carecenter.NurseIDInCharge;
?-------------------------------------------------------------------------
----------------------------------------
```

A Technician view showing the Technicians name, TechnicianID, PersonID, and their employeeID. This will tell exactly which technician is which.

```
CREATE VIEW technicianView AS
SELECT FirstName, LastName, TechnicianID, employee.EmployeeID
FROM Person, Technician, Employee
WHERE technician.EmployeeID = employee.EmployeeID AND employee.PersonID =
person.PersonID;
?-------------------------------------------------------------------------
----------------------------------------
```

An Employee view selecting the employee's information where the employee's PersonID correlates with personID.

```
CREATE VIEW employeeView AS
SELECT FirstName, LastName, EmployeeTypes, employee.EmployeeID
FROM Person, Employee
WHERE employee.PersonID = person.PersonID;
?-------------------------------------------------------------------------
----------------------------------------
```

A Volunteer view selecting the volunteer's info where the volunteer's personID correlates with personID.

```
CREATE VIEW volunteerView AS
SELECT FirstName, LastName, PersonRole, Skill, VolunteerID
FROM Person, Volunteer
WHERE volunteer.PersonID = person.PersonID;
```

?----------

A Patient view selecting the Patients info, as well as showing if the patient is in or out of the care center where the patient's PersonID correlates with person ID.

```
CREATE VIEW patientView AS
SELECT FirstName, LastName, isOutpatient, patientID
FROM Person, Patient, Outpatient
WHERE patient.PersonID = person.PersonID AND patient.Outpatient =
outpatient.Outpatient;
```

Replication

Master CMDS

Create a Master replication user

```
CREATE USER 'rep'@'10.10.1.1' INDENTIFIED BY 'password';
```

Grant the user privileges

```
GRANT ALL ON *.* TO 'rep'@'10.10.1.1' IDENTIFIED BY 'password';
```

Flush tables

```
FLUSH TABLES WITH READ LOCK;
```

Unlock tables

```
UNLOCK TABLES;
```

SLAVE CMDS

Stop the SLAVE

```
Stop slave;
```

Connect it to the MASTER server

Change master to

```
master_host='10.10.1.1',
master_user='rep',
master_password='password',
master_log_file='mysql-bin.000002',
master_log_pos=483,
master_connect_retry=10;
```

Start the SLAVE

```
Start slave;
```

See if everything is properly connected
Show slave status\G;

Now any database you've created on the MASTER server will be replicated over to the SLAVE server.

Backup & Restore

Backup

Command line
Mysqldump -u root -p a5 > a5_backup.sql

Restore

MYSQL
Create database a55;

Command line
Mysql -u root -p a55 > a5_backup.sql