**ITS 364**

**Lab 05 Stored Procedures**

**MYSQL**

**TEAM 04**

Michelle Gutierrez  
Jose Huizar

Dhruvkumar Patel

Jeremy Savage  
Donn Villaruz

**GROUPS – Michelle Gutierrez**

**# Insert new group into groups table**

CREATE PROCEDURE 'dmcs'.insertGroup' (IN v\_group\_name VARCHAR(45), IN v\_group\_desc VARCHAR(100), v\_group\_leader VARCHAR(45), IN v\_group\_specialization VARCHAR(45))

BEGIN

INSERT INTO GROUPS (group\_name, group\_desc, group\_leader, group\_specialization)

VALUES (v\_group\_name, v\_group\_desc, v\_group\_leader, v\_group\_specialization);

END

**# DELETE group based on group\_id**

CREATE PROCEDURE 'dmcs.deleteGroup' (IN v\_group\_id INT(11))

BEGIN

DELETE FROM GROUPS

WHERE group\_id = v\_group\_id;

END

**# UPDATE change in group leader based on group\_id**

CREATE PROCEDURE 'dmcs.updateGroup' (IN v\_group\_id INT(11), IN v\_group\_leader VARCHAR(45))

BEGIN

UPDATE GROUPS

SET group\_leader = v\_group\_leader

WHERE group\_id = v\_group\_id;

END

**# GET group name, group leader, group desc based on group\_id**

CREATE PROCEDURE 'dmcs.getGroup' (IN v\_group\_id(11))

BEGIN

SELECT groups.group\_name, groups.group\_desc, groups.group\_id, person.person\_id, person.person\_name

FROM GROUPS, PERSON

WHERE groups.group\_leader = person.person\_id

AND group\_id = v\_grp\_id;

END

**LOCATION – Michelle Gutierrez**

**# Insert new location in LOCATION table**

CREATE PROCEDURE 'dmcs'.insertLocation' (IN v\_location\_desc VARCHAR(45), IN v\_location\_address VARCHAR(80))

BEGIN

INSERT INTO LOCATION (location\_desc, location\_address)

VALUES (v\_location\_desc, v\_location\_address);

END

**# DELETE location based on location\_id**

CREATE PROCEDURE 'dmcs.deleteLocation' (IN v\_location\_id INT(11))

BEGIN

DELETE FROM LOCATION

WHERE location\_id = v\_location\_id;

END

**# UPDATE location address based on location\_id**

CREATE PROCEDURE 'dmcs.updateGroup' (IN v\_location\_id INT(11), IN v\_location\_address VARCHAR(80))

BEGIN

UPDATE LOCATION

SET location\_address = v\_location\_address

WHERE location\_id = v\_location\_id;

END

**# GET location address based on location\_desc**

CREATE PROCEDURE 'dmcs.getLocation' (IN v\_loc\_desc VARCHAR(45))

BEGIN

SELECT location\_id, location\_address

FROM LOCATION

WHERE location\_desc LIKE CONCAT("%", v\_loc\_desc, "%");

END

**GROUP\_LOCATION – Michelle Gutierrez**

**# Insert new group location record based on group\_id and location\_id**

CREATE PROCEDURE 'dmcs'.insertGroupLocation' (IN v\_group\_id INT(11), IN v\_task\_id INT(11))

BEGIN

INSERT INTO GROUP\_LOCATION (group\_id, location\_id)

VALUES (v\_group\_id, v\_location\_id);

END

**# DELETE group location based on group\_id**

CREATE PROCEDURE 'dmcs.deleteGroupLocation' (IN v\_group\_id INT(11))

BEGIN

DELETE FROM GROUP\_LOCATION

WHERE group\_id = v\_group\_id;

END

**# UPDATE change in location based on group\_id**

CREATE PROCEDURE 'dmcs.updateGroupLocation' (IN v\_group\_id INT(11), IN v\_location\_ID INT(11))

BEGIN

UPDATE GROUP\_LOCATION

SET location\_id = v\_location\_id

WHERE group\_id = v\_group\_id;

END

**# GET location of specified group (id)**

CREATE PROCEDURE `dmcs.getGroup\_Location(IN v\_grp\_id INT(11))

BEGIN

SELECT LOCATION.location\_address

FROM GROUP\_LOCATION, LOCATION

WHERE GROUP\_LOCATION.location\_id = LOCATION.location\_id

AND GROUP\_LOCATION.group\_id = v\_grp\_id;

END

**TASKS – Michelle Gutierrez**

**# Insert new task in TASKS table**

CREATE PROCEDURE 'dmcs'.insertTask' (IN v\_task\_desc VARCHAR(100), IN v\_project\_id INT(11))

BEGIN

INSERT INTO TASKS (task\_desc, project\_id)

VALUES (v\_task\_desc, v\_project\_id);

END

**# DELETE task based on task\_id**

CREATE PROCEDURE 'dmcs.deleteTask' (IN v\_task\_id INT(11))

BEGIN

DELETE FROM TASKS

WHERE task\_id = v\_task\_id;

END

**# UPDATE task description based on task\_id**

CREATE PROCEDURE 'dmcs.updateTask' (IN v\_proj\_id INT(11),IN v\_task\_id INT(11), IN v\_task\_desc VARCHAR(100))

BEGIN

UPDATE TASKS

SET task\_desc = v\_task\_desc

WHERE project\_id = v\_proj\_id

AND task\_id = v\_task\_id;

END

**# GET list of tasks and respective group info for specified project (id)**

CREATE PROCEDURE 'dmcs.getTask' (IN v\_task\_id INT(11))

BEGIN

SELECT tasks.task\_id, tasks.task\_desc, groups.group\_id, groups.group\_name, groups.group\_leader

FROM group\_tasks, groups, tasks, project

WHERE group\_tasks.task\_id = tasks.task\_id

AND group\_tasks.group\_id = groups.group\_id

AND tasks.project\_id = project.project\_id

AND project.project\_id = v\_proj\_id;

END

**GROUP\_TASKS – Michelle Gutierrez**

**#Insert new group record based on group\_id and task\_id**

CREATE PROCEDURE 'dmcs'.insertGroupTasks' (IN v\_group\_id INT(11), IN v\_task\_id INT(11))

BEGIN

INSERT INTO GROUP\_TASKS (group\_id, task\_id)

VALUES (v\_group\_id, v\_task\_id);

END

**#DELETE row based on group\_id**

CREATE PROCEDURE 'dmcs.deleteGroupTasks' (IN v\_group\_id INT(11))

BEGIN

DELETE FROM GROUP\_TASKS

WHERE group\_id = v\_group\_id;

END

**#UPDATE change in task\_id on group\_id**

CREATE PROCEDURE 'dmcs.updateGroupTasks' (IN v\_group\_id INT(11), IN v\_task\_id INT(11))

BEGIN

UPDATE GROUPS

SET task\_id = v\_task\_id

WHERE group\_id = v\_group\_id;

END

**# GET tasks associated with group\_id**

CREATE PROCEDURE 'dmcs.getGroupTasks' (IN v\_grp\_id INT(11))

BEGIN

SELECT group\_tasks.task\_id, tasks.task\_desc

FROM group\_tasks, tasks, groups

WHERE group\_tasks.task\_id = tasks.task\_id

AND group\_tasks.group\_id = groups.group\_id

AND group\_tasks.group\_id = v\_grp\_id;

END

CREATE PROCEDURE 'dmcs.getGroupTasks2' (IN v\_task\_id INT(11))

BEGIN

-- Get group info (id, name) by specified task (id)

SELECT group\_tasks.task\_id, group\_tasks.group\_id, groups.group\_name

FROM group\_tasks, groups

WHERE group\_tasks.group\_id = groups.group\_id

AND group\_tasks.task\_id = v\_task\_id;

END

**PROJECT – Jose Huizar**

**#INSERT new record based on project\_id, project\_name, project\_desc**

CREATE PROCEDURE 'dmcs'.'insProj' (IN v\_proj\_id INT(11), IN v\_proj\_name VARCHAR(45), IN v\_proj\_desc VARCHAR(100))

BEGIN

INSERT INTO PROJECT (project\_id, project\_name, project\_desc)

VALUES (v\_proj\_id, v\_proj\_name, v\_proj\_desc)

END

**#DELETE record based on project\_id, project\_name, project\_desc**

CREATE PROCEDURE 'dmcs'.'delProj' (IN v\_proj\_id INT(11))

BEGIN

DELETE FROM PROJECT

WHERE project\_id = v\_proj\_id;

END

**#UPDATE record based on project\_id**

CREATE PROCEDURE 'dmcs'.'updProj' (IN v\_proj\_id INT(11), IN v\_proj\_name VARCHAR(45))

BEGIN

UPDATE PROJECT

SET project\_desc = v\_proj\_desc

WHERE project\_id = v\_proj\_id;

END

**#GET all projects and info by specific disaster id**

CREATE PROCEDURE 'dmcs'.'getProj' (IN v\_disaster\_id INT(11))

BEGIN

SELECT project.project\_id, project.project\_name, project.project\_desc, requests.request\_status

FROM project, disaster, encounter, requests

WHERE encounter.project\_id = project.project\_id

AND encounter.disaster\_id = disaster.disaster\_id

AND encounter.request\_id = requests.request\_id

AND encounter.disaster\_id = v\_disaster\_id

ORDER BY project.project\_id DESC;

END

**PROJECT\_LOCATION – Jose Huizar**

**#INSERT new record based on project\_id**

CREATE PROCEDURE 'dmcs'.'insProj\_loc' (IN v\_proj\_id INT(11))

BEGIN

INSERT INTO PROJECT\_LOCATION (project\_id)

VALUES (v\_proj\_id)

END

**#DELETE record based on project\_id**

CREATE PROCEDURE 'dmcs'.'delProj\_loc' (IN v\_proj\_id INT(11))

BEGIN

DELETE FROM PROJECT\_LOCATION

WHERE proejct\_id = v\_proj\_id;

END

**#UPDATE record based on project\_id, location\_id**

CREATE PROCEDURE 'dmcs'.'updProj\_loc' (IN v\_proj\_id INT(11), IN v\_loc\_id INT (11))

BEGIN

UPDATE PROJECT\_LOCATION

SET project\_id = v\_proj\_id

WHERE location\_id = v\_loc\_id;

END

**#GET new record based on project\_id, location\_id**

CREATE PROCEDURE 'dmcs'.'getProj\_loc' (IN v\_loc\_id INT (11))

BEGIN

SELECT PROJECT\_LOCATION.project\_id, PROJECT.project\_name, PROJECT.project\_desc

FROM PROJECT, PROJECT\_LOCATION, LOCATION

WHERE PROJECT.project\_id = PROJECT\_LOCATION.project\_id

AND PROJECT\_LOCATION.location\_id = LOCATION.location\_id

AND PROJECT\_LOCATION.location\_id = v\_loc\_id;

END

CREATE PROCEDURE 'dmcs'.'getProj\_loc2' (IN v\_proj\_id INT (11))

BEGIN

-- Get project info (name, desc, location) from specified proj id

SELECT project.project\_name, project.project\_desc, location.location\_address

FROM PROJECT, PROJECT\_LOCATION, LOCATION

WHERE PROJECT\_LOCATION.project\_id = project.project\_id

AND PROJECT\_LOCATION.location\_id = LOCATION.location\_id

AND project.project\_id = v\_proj\_id;

END

**PRODUCT – Michelle Gutierrez**

**# INSERT: add new record by providing product\_type & product\_quantity**

CREATE PROCEDURE `dmcs`.`insertProduct` (IN v\_pro\_type VARCHAR(45), v\_pro\_qty INT(11))

BEGIN

INSERT INTO PRODUCT (product\_type, product\_quantity)

VALUES (v\_pro\_type, v\_pro\_qty);

END

**# DELETE: remove a product’s record by product\_id**

CREATE PROCEDURE `dmcs`.`deleteProduct` (IN v\_pro\_id INT(11))

BEGIN

DELETE FROM PRODUCT

WHERE product\_id = v\_pro\_id;

END

**# UPDATE: change product quantity for a product using product\_id**

CREATE PROCEDURE `dmcs`.`updateProduct` (IN v\_pro\_id INT(11), IN v\_pro\_qty INT(11))

BEGIN

UPDATE PRODUCT

SET product\_quantity = v\_pro\_qty

WHERE product\_id = v\_pro\_id;

END

**# GET: display quantity of a product using product\_id**

CREATE PROCEDURE `dmcs`.`getProduct` (IN v\_pro\_id INT(11))

BEGIN

SELECT product\_quantity

FROM PRODUCT

WHERE product\_id = v\_pro\_id;

END

**WAREHOUSE – Michelle Gutierrez**

**# INSERT: add new record by providing warehouse\_name & location\_id**

CREATE PROCEDURE `dmcs`.`insertWarehouse` (IN v\_wh\_name VARCHAR(45), IN v\_loc\_id INT(11))

BEGIN

INSERT INTO WAREHOUSE (warehouse\_name, location\_id)

VALUES (v\_wh\_name, v\_loc\_id);

END

**# DELETE: remove a warehouse’s record by warehouse\_id**

CREATE PROCEDURE `dmcs`.`deleteWarehouse` (IN v\_wh\_id INT(11))

BEGIN

DELETE FROM WAREHOUSE

WHERE warehouse\_id = v\_wh\_id;

END

**# UPDATE: change warehouse name for a warehouse using warehouse\_id**

CREATE PROCEDURE `dmcs`.`updateWarehouse` (IN v\_wh\_id INT(11), IN v\_wh\_name VARCHAR(45))

BEGIN

UPDATE WAREHOUSE

SET warehouse\_name = v\_wh\_name

WHERE warehouse\_id = v\_wh\_id;

END

**# GET: display location of warehouse using warehouse\_id**

CREATE PROCEDURE `dmcs`.`getWarehouse` (IN v\_wh\_id INT(11))

BEGIN

SELECT location\_id

FROM WAREHOUSE

WHERE warehouse\_id = v\_wh\_id;

END

**WAREHOUSE\_PRODUCT – Jose Huizar**

**#INSERT new record based on warehouse\_id**

CREATE PROCEDURE 'dmcs'.'insWH\_P' (IN v\_wh\_id INT(11))

BEGIN

INSERT INTO WAREHOUSE\_PRODUCT (warehouse\_id)

VALUES (v\_wh\_id)

END

**#DELETE record based on warehouse\_id**

CREATE PROCEDURE 'dmcs'.'delWH\_P' (IN v\_wh\_id INT(11))

BEGIN

DELETE FROM WAREHOUSE\_PRODUCT

WHERE warehouse\_id = v\_wh\_id;

END

**#UPDATE record based on warehouse\_id, product\_id**

CREATE PROCEDURE 'dmcs'.'updWH\_P' (IN v\_wh\_id INT(11), IN v\_prod\_id INT (11))

BEGIN

UPDATE WAREHOUSE\_PRODUCT

SET warehouse\_id = v\_wh\_id

WHERE product\_id = v\_prod\_id;

END

**#GET new record based on warehouse\_id**

CREATE PROCEDURE 'dmcs'.'getWH\_P' (IN v\_wh\_id INT(11))

BEGIN

-- Get warehouse info (id, name, address) of a specified product

SELECT WAREHOUSE.warehouse\_id, WAREHOUSE.warehouse\_name, LOCATION.location\_address

FROM WAREHOUSE\_PRODUCT, WAREHOUSE, LOCATION

WHERE WAREHOUSE\_PRODUCT.warehouse\_id = WAREHOUSE.warehouse\_id

AND WAREHOUSE.location\_id = LOCATION.location\_id

AND WAREHOUSE\_PRODUCT.product\_id = v\_prod\_id;

END

**PERSON – Dhruvkumar Patel**

**# Insert a new record based on person\_id, name, address, phone and email**

CREATE PROCEDURE insertPerson (IN v\_id INT (11), IN v\_name VARCHAR(50), IN v\_add VARCHAR(8), IN v\_ph VARCHAR(10), IN v\_email VARCHAR (100))

BEGIN

INSERT INTO person (person\_id, person\_name, person\_address, person\_phone, person\_email)

VALUES(v\_id, v\_name, v\_add, v\_ph, v\_email);

END

**#Deleting a record based on person\_id**

CREATE PROCEDURE deletePerson (IN v\_id INT (11))

BEGIN

DELETE FROM person

WHERE person\_id = v\_id;

END

**# Updating Persons Phones Number based on Person\_Name**

CREATE PROCEDURE updatePerson (IN v\_name VARCHAR(50), IN v\_newPhone VARCHAR (10))

BEGIN

UPDATE person

SET person\_phone = v\_newPhone

WHERE person\_name = v\_name;

END

**# Getting Person info based on their email**

CREATE PROCEDURE getPerson (IN v\_prsn\_email VARCHAR (50))

BEGIN

SELECT \*

FROM person

WHERE person\_email = v\_prsn\_email;

END

CREATE PROCEDURE getPerson2 (IN v\_disas\_id INT (11), IN v\_req\_id INT (11);)

BEGIN

-- Get person info based on his/her request id

SELECT person.person\_id, person.person\_name, person.person\_address, person.person\_phone, person.person\_email

FROM person, requests, encounter

WHERE encounter.person\_id = person.person\_id

AND encounter.request\_id = requests.request\_id

AND encounter.disaster\_id = v\_disas\_id

AND encounter.request\_id = v\_req\_id;

END

**EMPLOYEE – Dhruvkumar Patel**

**#Insert a new Employee record based on person\_id, job\_desc and department**

CREATE PROCEDURE insertEmployee (IN v\_emp\_id INT (11), IN v\_emp\_job\_desc VARCHAR(100), IN v\_emp\_dept VARCHAR(45))

BEGIN

INSERT INTO employee (person\_id, job\_desc, department)

VALUES(v\_emp\_id, v\_emp\_job\_desc, v\_emp\_dept);

END

**#Deleting an Employee's record based on person\_id**

CREATE PROCEDURE deleteEmployee (IN v\_emp\_id INT (11))

BEGIN

DELETE FROM employee

WHERE person\_id = v\_emp\_id;

END

**#Updating Employee's job description based on person\_id**

CREATE PROCEDURE updateEmployee (IN v\_emp\_id INT(11), IN v\_emp\_job\_desc VARCHAR (100))

BEGIN

UPDATE employee

SET job\_desc = v\_emp\_job\_desc

WHERE person\_id = v\_emp\_id;

END

**#Getting All employees in a department**

CREATE PROCEDURE getEmployee (IN v\_emp\_dept VARCHAR (50))

BEGIN

SELECT person\_name, person\_email, department

FROM person, employee

WHERE person.person\_id = employee.person\_id

AND department = v\_emp\_dept;

END

BEGIN

-- Get all employees (id, name, email, dept, job\_desc)

SELECT employee.person\_id, person.person\_name, person.person\_email, employee.department, employee.job\_desc

FROM person, employee

WHERE person.person\_id = employee.person\_id;

END

**DONOR – Dhruvkumar Patel**

**#Insert a new Donor record based on person\_id, donation\_type and donation\_amount**

CREATE PROCEDURE insertDonor (IN v\_don\_id INT (11), IN v\_don\_type VARCHAR(100), IN v\_don\_amt INT(11))

BEGIN

INSERT INTO donor (person\_id, donation\_type, donation\_amount)

VALUES(v\_don\_id, v\_don\_type, v\_don\_amt);

END

**#Deleting a Donor's record based on person\_id**

CREATE PROCEDURE deleteDonor (IN v\_don\_id INT (11))

BEGIN

DELETE FROM donor

WHERE person\_id = v\_don\_id;

END

**#Updating donation\_amount based on person\_name**

CREATE PROCEDURE updateDonor (IN v\_don\_name VARCHAR(25), IN v\_don\_amount INT (11))

BEGIN

UPDATE donor, person

SET donation\_amount = v\_don\_amount

WHERE person.person\_id = donor.person\_id

AND person.person\_name = v\_don\_name;

END

**#Getting donation\_amount based on donation\_type**

CREATE PROCEDURE getQuantity (IN v\_don\_type VARCHAR (25))

BEGIN

SELECT donation\_amount

FROM donor

WHERE donation\_type = v\_don\_type;

END

**CLIENT – Dhruvkumar Patel**

**#Insert a new record based on person\_id, client\_desc, client\_status**

CREATE PROCEDURE insertClient (IN v\_client\_id INT (11), IN v\_client\_desc VARCHAR(150), IN v\_client\_status VARCHAR(25))

BEGIN

INSERT INTO client (person\_id, client\_desc, client\_status)

VALUES(v\_client\_id, v\_client\_desc, v\_client\_status);

END

**#Deleting a Clients's record based on person\_id**

CREATE PROCEDURE deleteClient (IN v\_client\_id INT (11))

BEGIN

DELETE FROM client

WHERE person\_id = v\_client\_id;

END

**#Updating client\_status based on person\_id**

CREATE PROCEDURE updateStatus (IN v\_client\_id INT(11), IN v\_client\_status VARCHAR(30))

BEGIN

UPDATE client

SET client\_status = v\_client\_status

WHERE person\_id = v\_client\_id;

END

**#Getting client\_names based on client\_status**

CREATE PROCEDURE getClient(IN v\_client\_status VARCHAR (25))

BEGIN

SELECT person\_name, client\_status

FROM person, client

WHERE person.person\_id = client.person\_id

AND client\_status = v\_client\_status;

END

**DISASTER – Dhruvkumar Patel**

**#Insert a new record based on disater\_id**

CREATE PROCEDURE insertDisaster (IN v\_disaster\_id INT (11), IN v\_disaster\_desc VARCHAR(100), IN v\_disaster\_date DATE, IN v\_disaster\_name VARCHAR(30))

BEGIN

INSERT INTO disaster (disaster\_id, disaster\_desc, disaster\_date, disaster\_name)

VALUES(v\_disaster\_id, v\_disaster\_desc, v\_disaster\_date, v\_disaster\_name);

END

**#Deleting a Disaster record based on disaster\_id**

CREATE PROCEDURE deleteDisaster (IN v\_disaster\_id INT (11))

BEGIN

DELETE FROM disaster

WHERE disaster\_id = v\_disaster\_id;

END

**#Updating disaster\_desc based on disaster\_id**

CREATE PROCEDURE updateDisaster (IN v\_disaster\_id INT(11), IN v\_disaster\_desc VARCHAR(80))

BEGIN

UPDATE disaster

SET disaster\_desc = v\_disaster\_status

WHERE disaster\_id = v\_disaster\_id;

END

**#Getting Disaster Information from Disaster Name**

CREATE PROCEDURE getDisaster(IN v\_disaster\_name VARCHAR (30))

BEGIN

SELECT disaster\_desc, disaster\_date, disaster\_name

FROM disaster

WHERE disaster\_name = v\_disaster\_name;

END

**REQUESTS – Jeremy Savage**

**#Delete Request**

CREATE PROCEDURE `deleteRequest`(IN v\_req\_id INT(11))

BEGIN

    DELETE FROM requests

    WHERE request\_id = v\_req\_id;

END

**#Add Request**

CREATE PROCEDURE `insertRequest`(IN v\_request\_time DATETIME, IN v\_request\_desc VARCHAR(80), IN v\_request\_status VARCHAR(30))

BEGIN

INSERT INTO requests(request\_time, request\_desc, request\_status) VALUES (v\_request\_time, v\_request\_status, v\_request\_status);

END

**#Update Request**

CREATE PROCEDURE `updateRequest`(IN req\_id INT(11), IN req\_status VARCHAR(30))

BEGIN

UPDATE requests

    SET request\_status = v\_req\_status

    WHERE request\_id = v\_req\_id;

END

**#Get Requests that are open**

CREATE PROCEDURE `getRequest`(IN v\_req\_status VARCHAR(30))

BEGIN

SELECT \*

FROM requests

WHERE request\_status = v\_req\_status;

END

BEGIN

-- Get all requests by specified disaster id

SELECT requests.request\_id, requests.request\_time, requests.request\_type, requests.request\_desc, requests.request\_status, person.person\_name

FROM encounter, requests, person

WHERE encounter.request\_id = requests.request\_id

AND encounter.person\_id = person.person\_id

AND encounter.disaster\_id = v\_disas\_id

ORDER BY request\_time DESC;

END

BEGIN

-- Get request info by specified disaster id and request id

SELECT requests.request\_id, requests.request\_time, requests.request\_type, requests.request\_desc, requests.request\_status

FROM requests, encounter

WHERE encounter.request\_id = requests.request\_id

AND encounter.disaster\_id = v\_disas\_id

AND encounter.request\_id = v\_req\_id;

END

**SOCIAL\_MEDIA – Jeremy Savage**

**#Delete Social Media by email**

CREATE PROCEDURE `deleteSocial\_Media`(in v\_sm\_email VARCHAR (60))

BEGIN

DELETE FROM social\_media

WHERE sm\_email = v\_sm\_email;

END

**#Get Social Media by ID**

CREATE PROCEDURE `getSocial\_Media`(in v\_req\_id INT (11))

BEGIN

SELECT sm\_email

FROM social\_media

WHERE request\_id = v\_req\_id;

END

**#Insert social media**

CREATE PROCEDURE `insertSocial\_Media`(in v\_req\_id INT (11), in v\_sm\_email VARCHAR (80))

BEGIN

INSERT INTO social\_media(request\_id, sm\_email) VALUES (v\_req\_id, v\_sm\_email);

END

**#Update social media**

CREATE PROCEDURE `updateSocial\_Media`(in v\_req\_id INT (11), in v\_sm\_email VARCHAR (60))

BEGIN

UPDATE social\_media

SET sm\_email = v\_sm\_email

WHERE request\_id = v\_req\_id;

END

**PHONE\_CALLS – Jeremy Savage**

**#delete phone calls by request id**

CREATE PROCEDURE `deletePhone\_calls`(IN v\_req\_id INT(11) )

BEGIN

DELETE FROM phone\_calls

WHERE request\_id = v\_req\_id;

END

**#get phone calls based on request id**

CREATE PROCEDURE `getPhone\_calls`(IN req\_id INT(11))

BEGIN

SELECT phone\_num

FROM phone\_calls

WHERE request\_id = req\_id;

END

**#insert phone calls**

CREATE PROCEDURE `insertPhone\_Calls`(IN req\_id INT(11), IN phone\_num VARCHAR(10))

BEGIN

INSERT INTO phone\_calls (request\_id, phone\_num)

VALUES (req\_id, phone\_num);

END

**#update phone number**

CREATE PROCEDURE `updatePhone\_Calls`(IN req\_id INT(11), IN pho\_num VARCHAR(10))

BEGIN

UPDATE phone\_calls

SET phone\_num = pho\_num

WHERE request\_id = req\_id;

END

**WEBSITE – Jeremy Savage**

**#delete website**

CREATE PROCEDURE `deleteWebsite`(in v\_web\_username VARCHAR (45))

BEGIN

DELETE FROM website

WHERE web\_username = v\_web\_username;

END

**#get website**

CREATE PROCEDURE `getWebsite`(in v\_req\_id INT (11))

BEGIN

SELECT web\_username

FROM website

WHERE request\_id = v\_req\_id;

END

**#insert website**

CREATE PROCEDURE `insertWebsite`(in v\_req\_id INT (11), in v\_web\_username VARCHAR (45))

BEGIN

insert into website(request\_id, web\_username) values(v\_req\_id, v\_web\_username);

END

**#update website username**

CREATE PROCEDURE `updateWebsite`(in v\_req\_id INT (11), in v\_web\_username VARCHAR (45))

BEGIN

UPDATE website

SET web\_username = v\_web\_username

WHERE request\_id = v\_req\_id;

END

**PRODUCT\_REQUESTS – Jeremy Savage**

**#delete product requests**

CREATE PROCEDURE `deleteProduct\_Requests`(in v\_product\_id INT (11))

BEGIN

DELETE FROM product\_requests

WHERE product\_id = v\_product\_id;

END

**#get product requests by request id**

CREATE PROCEDURE `getProduct\_Requests`(in v\_request\_id INT (11))

BEGIN

SELECT product\_id

FROM product\_request

WHERE request\_id = v\_request\_id;

END

**#insert product request**

CREATE PROCEDURE `insertProduct\_Requests`(in v\_product\_id INT (11), in v\_request\_id INT (11))

BEGIN

INSERT INTO product\_requests(product\_id, request\_id) VALUES(v\_product\_id, v\_request\_id);

END

**#update product\_request product id**

CREATE PROCEDURE `updateProduct\_Requests`(in v\_product\_id INT (11), in v\_request\_id INT (11))

BEGIN

UPDATE product\_requests

SET product\_id = v\_product\_id

WHERE request\_id = v\_request\_id;

END

**VOLUNTEER – Donn Villaruz**

**# INSERT: add new record by providing person\_id & availability**

CREATE PROCEDURE `dmcs`.`insertVolunteer` (IN v\_vol\_id INT(11), IN v\_new\_time VARCHAR(50))

BEGIN

INSERT INTO VOLUNTEER (person\_id, availability)

VALUES (v\_vol\_id, v\_new\_time);

END

**# DELETE: remove a volunteer’s record by person\_id**

CREATE PROCEDURE `dmcs`.`deleteVolunteer` (IN v\_vol\_id INT(11))

BEGIN

DELETE FROM VOLUNTEER

WHERE person\_id = v\_vol\_id;

END

**# UPDATE: change availability for a volunteer using person\_id**

CREATE PROCEDURE `dmcs`.`updateVolunteer` (IN v\_vol\_id INT(11), IN v\_change\_time VARCHAR(50))

BEGIN

UPDATE VOLUNTEER

SET availability = v\_change\_time

WHERE person\_id = v\_vol\_id;

END

**# GET: all volunteer information by id**

CREATE PROCEDURE `dmcs`.`getVolunteer` (IN v\_vol\_id INT(11))

BEGIN

SELECT volunteer.person\_id, person.person\_name, volunteer.availability

FROM VOLUNTEER, PERSON

WHERE person.person\_id = volunteer.person\_id;

END

**SKILLS – Donn Villaruz**

**# INSERT record based on person\_id and skill\_id**

CREATE PROCEDURE `dmcs`.`spINS\_vol\_skills` (IN vol\_id INT(11), IN skill\_no INT(11))

BEGIN

INSERT INTO VOLUNTEER\_SKILLS (person\_id, skill\_id)

VALUES (vol\_id, skill\_no);

END

**# DELETE row based on person\_id**

CREATE PROCEDURE `dmcs`.`spDEL\_vol\_skills` (IN vol\_id INT(11))

BEGIN

DELETE FROM VOLUNTEER\_SKILLS

WHERE person\_id = vol\_id;

END

**# UPDATE skill\_id based on person\_id**

CREATE PROCEDURE `dmcs`.`spUPD\_vol\_skills` (IN vol\_id INT(11), IN skill\_no INT(11))

BEGIN

UPDATE VOLUNTEER\_SKILLS

SET skill\_id = skill\_no

WHERE person\_id = vol\_id;

END

**# GET list of skills based on person\_id**

CREATE PROCEDURE `dmcs`.`spGET\_vol\_skills` (IN v)keyword VARCHAR(50))

BEGIN

SELECT \* FROM SKILLS

WHERE skill\_desc LIKE CONCAT("%", v\_keyword, "%");

END

**VOLUNTEER\_GROUPS – Donn Villaruz**

**# INSERT assign a volunteer to a group**

CREATE PROCEDURE `dmcs`.`spINS\_vol\_grps` (IN vol\_id INT(11), IN grp\_id INT(11))

BEGIN

INSERT INTO VOLUNTEER\_GROUPS (person\_id, group\_id)

VALUES (person\_id, grp\_id);

END

**# DELETE remove a volunteer from all assigned groups**

CREATE PROCEDURE `dmcs`.`spINS\_vol\_grps` (IN vol\_id INT(11))

BEGIN

DELETE FROM VOLUNTEER\_GROUPS

WHERE person\_id = vol\_id;

END

**# UPDATE switch a volunteer to another group**

CREATE PROCEDURE `dmcs`.`spINS\_vol\_grps` (IN vol\_id INT(11), IN grp\_id INT(11))

BEGIN

UPDATE VOLUNTEER\_GROUPS

SET group\_id = grp\_id

WHERE person\_id = vol\_id;

END

**# GET all volunteers for a specified group**

CREATE PROCEDURE `dmcs`.`spINS\_vol\_grps` (IN vol\_id INT(11))

BEGIN

SELECT volunteer\_groups.group\_id, volunteer\_groups.person\_id, person.person\_name

FROM volunteer\_groups, person

WHERE volunteer\_groups.person\_id = person.person\_id

AND group\_id = v\_group\_id;

END

**VOLUNTEER\_TASKS – Donn Villaruz**

**# INSERT assign new task to volunteer**

CREATE PROCEDURE `dmcs`.`spINS\_vol\_tasks` (IN vol\_id INT(11), IN tsk\_id INT(11))

BEGIN

INSERT INTO VOLUNTEER\_TASKS (person\_id, task\_id)

VALUES (vol\_id, tk\_id);

END

**# DELETE all assigned tasks to volunteer**

CREATE PROCEDURE `dmcs`.`spDEL\_vol\_tasks` (IN vol\_id INT(11))

BEGIN

DELETE FROM VOLUNTEER\_TASKS

WHERE person\_id = vol\_id;

END

**# UPDATE assign a diff task to volunteer**

CREATE PROCEDURE `dmcs`.`spUPD\_vol\_tsks` (IN vol\_id INT(11), IN tsk\_id INT(11))

BEGIN

UPDATE VOLUNTEER\_TASKS

SET task\_id = tsk\_id;

WHERE person\_id = vol\_id;

END

**# GET list of tasks assigned to volunteer**

CREATE PROCEDURE `dmcs`.`spGET\_vol\_tsks` (IN vol\_id INT(11))

BEGIN

SELECT \* FROM VOLUNTEER\_TASKS

WHERE person\_id = vol\_id;

END

**VOLUNTEER\_SKILLS – Donn Villaruz**

**# INSERT record based on person\_id and skill\_id**

CREATE PROCEDURE `dmcs`.`spINS\_vol\_skills` (IN vol\_id INT(11), IN skill\_no INT(11))

BEGIN

INSERT INTO VOLUNTEER\_SKILLS (person\_id, skill\_id)

VALUES (vol\_id, skill\_no);

END

**# DELETE row based on person\_id**

CREATE PROCEDURE `dmcs`.`spDEL\_vol\_skills` (IN vol\_id INT(11))

BEGIN

DELETE FROM VOLUNTEER\_SKILLS

WHERE person\_id = vol\_id;

END

**# UPDATE skill\_id based on person\_id**

CREATE PROCEDURE `dmcs`.`spUPD\_vol\_skills` (IN vol\_id INT(11), IN skill\_no INT(11))

BEGIN

UPDATE VOLUNTEER\_SKILLS

SET skill\_id = skill\_no

WHERE person\_id = vol\_id;

END

**# GET list of skills based on person\_id**

CREATE PROCEDURE `dmcs`.`spGET\_vol\_skills` (IN vol\_id INT(11))

BEGIN

SELECT \* FROM VOLUNTEER\_SKILLS

WHERE person\_id = vol\_id;

END

**ENCOUNTER – Donn Villaruz**

**# INSERT encounter**

CREATE PROCEDURE `dmcs`.`spInsertEncounter ` (IN disas\_id INT(11), IN v\_proj\_id INT(11), IN v\_req\_id INT (11), IN v\_prsn\_id INT(11), OUT last\_enc INT (11))

BEGIN

-- Create new encounter

INSERT INTO encounter (disaster\_id, project\_id, request\_id, person\_id)

VALUES (v\_disas\_id, v\_proj\_id, v\_req\_id, v\_prsn\_id);

SET last\_enc\_id = LAST\_INSERT\_ID();

END

**# update encounter**

CREATE PROCEDURE `dmcs`.`spInsertEncounter ` (IN disas\_id INT(11), IN v\_proj\_id INT(11), IN v\_req\_id INT (11), IN v\_prsn\_id INT(11))

BEGIN

-- Assign a project to an encounter

UPDATE encounter

SET project\_id = v\_proj\_id

WHERE disaster\_id = v\_disas\_id

AND request\_id = v\_req\_id

AND person\_id = v\_prsn\_id;

END