CPSC 5021 In-Class Exercise (Join, Trigger)

1. Using the EMPLOYEE, JOB, and PROJECT tables, write the SQL code that will produce the results shown below.

PROJ_NAME	PROJ_VALUE	PROJ_BALANCE	EMP_LNAME	EMP_FNMAE	EMP_INITIAL	JOB_CODE	JOB_DESCRIPTION	JOB_CHG_HOUR
Rolling Tide	805000.00	500345.20	Senior	David	H	501	Systems Analyst	96.75
Evergreen	1454500.00	1002350.00	Arbough	June	E	500	Programmer	35.75
Starflight	2650500.00	2309880.00	Alonzo	Maria	D	501	Systems Analyst	96.75
Amber Wave	3500500.00	2110346.00	Washington	Ralph	B	501	Systems Analyst	96.75

```
select proj_name, proj_value, proj_balance, employee.emp_lname, emp_fname, emp_initial,
employee.job_code, job.job_description, job.job_chg_hour
from project, employee, job
where employee.emp_num = project.emp_num
and job.job_code = employee.job_code;
```

2. Using the data in the ASSIGNMENT table, write the code that will yield the total number of hours worked for each employee and the total charges stemming from those hours worked. The results of running that query are shown below.

EMP_NUM	EMP_LNAME	SumOfASSIGN_HOURS	SumOfASSIGN_CHARGE
101	News	3.1	387.50
102	Senior	10.1	904.90
103	Arbough	10.5	887.25
104	Ramoras	2.8	270.90
105	Johnson	8.2	931.00
107	Alonzo	4.3	451.50
108	Washington	8.3	840.15

```
select assignment.emp_num, employee.emp_lname, sum(assignment.assign_hours) as
sumofassign_hours, sum(assignment.assign_chg_hr * assignment.assign_hours) as
sumofassign_charge
from employee, assignment
where employee.emp_num = assignment.emp_num
group by assignment.emp_num, employee.emp_lname
```

3. Create a trigger in MySQL to audit the changes of the *employees* table.

Step 1: First, create a table *employees* using the following structure. Insert 2 records in this table. You can pick any values, as long as they are of the correct type as specified below.

employees

ATTRIBUTE NAME	ATTRIBUTE TYPE
employeeNumber	int(11)
lastName	varchar(50)
firstName	varchar(50)
extension	varchar(10)
email	varchar(100)
officeCode	varchar(10)
reportsTo	int(11)
jobTitle	varchar(50)

Step 2: create a new table named employees_audit to keep the changes of the employee records using the following structure.

employees_audit

ATTRIBUTE NAME	ATTRIBUTE TYPE
id	int(11)
employeeNumber	int(11)
lastName	varchar(50)
changedOn	datetime
action	varchar(50)

Step 3: create a BEFORE UPDATE trigger to be invoked before a change is made to the *employees* table. Before a change is made to the *employees* table, the old employeeNumber , the old lastName, the time of change must be inserted in the table "employees_audit". The action is set to 'update'.

```
DELIMITER $$

CREATE TRIGGER before_employee_update
   BEFORE UPDATE ON employees
   FOR EACH ROW BEGIN
   INSERT INTO employees_audit
   SET action = 'update',
       employeeNumber = OLD.employeeNumber,
       lastName = OLD.lastName,
      changedOn = NOW();
END$$
```

Step 4: Update an employee record to test if the trigger is really invoked.

Step 5: Create a trigger “employee_insert” on table “employees”. The trigger is fired after a row is inserted in table “employees”. After a row is inserted in table “employees”, the “employeeNumber”, “lastName” , the insertion time, and the action is inserted in table “employees_audit”. The action is set to ‘insert’. The “id” is set to 2.

```
delimiter $$
CREATE TRIGGER employee_insert
AFTER insert ON employees
FOR EACH ROW
BEGIN
insert into employees_audit values(2, new.employeeNumber,
new.lastName,NOW(),'insert');
END$$
```

Step 6: Insert an employee record to test if the trigger is really fired.