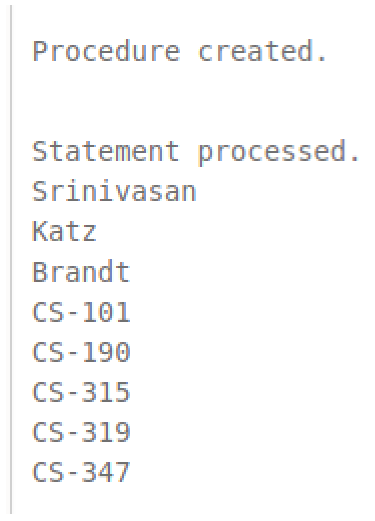
Parth Shukla
Lab 8
190905104

Procedures:
1. Based on the University Database Schema in Lab 2, write a procedure which takes the dept_name as input parameter and lists all the instructors associated with the department as well as list all the courses offered by the department. Also, write an anonymous block with the procedure call.

```
Create or replace procedure dept_count(dep_name in varchar) is
 cursor curseinst is select * from instructor where instructor.dept_name=dept_count.dep_name;
 cursor cursecourse is select * from course where course.dept_name = dept_count.dep_name;
begin
for row in curseinst loop
dbms_output.put_line(row.name);
end loop;

for row in cursecourse loop
dbms_output.put_line(row.course_id);
end loop;
end;
/

Declare
begin
dept_count('Comp. Sci.');
end;
/
```



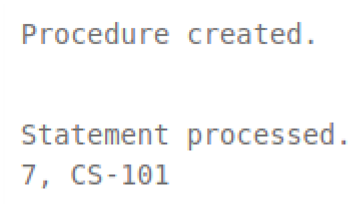
```
Procedure created.


Statement processed.
Srinivasan
Katz
Brandt
CS-101
CS-190
CS-315
CS-319
CS-347
```

2. Based on the University Database Schema in Lab 2, write a Pl/Sql block of code that lists the most popular course (highest number of students take it) for each of the departments. It should make use of a procedure course_popular which finds the most popular course in the given department.

```
create or replace procedure course_popular(dept_name in varchar) is
cursor cursecourse is select * from course where course.dept_name =
course_popular.dept_name;
max_count integer;
temp integer;
max_course varchar(20);
begin
max_count := 0;

for row in cursecourse loop
select count(*) into temp from takes where takes.course_id = row.course_id;
if temp > max_count then
max_count := temp;
max_course := row.course_id;
end if;
end loop;
dbms_output.put_line(max_count || ', ' || max_course);
end;
/
```



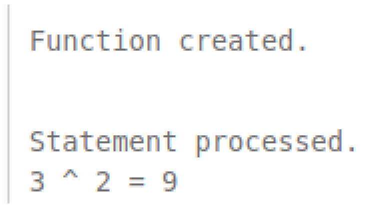
```
Procedure created.


Statement processed.
7, CS-101
```

Functions:
3. Write a function to return the Square of a given number and call it from an anonymous block.

```
CREATE OR REPLACE FUNCTION square (x number)
        RETURN number AS
                s number;
BEGIN
        s := x * x;
        RETURN s;
END;
/
```

```
DECLARE
BEGIN
	dbms_output.put_line('3 ^ 2 = ' || square(3));
END;
/
```



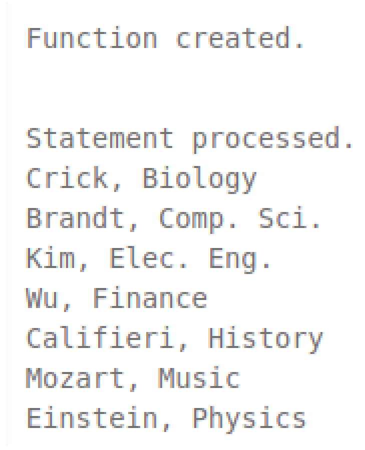
```
Function created.


Statement processed.
3 ^ 2 = 9
```

4. Based on the University Database Schema in Lab 2, write a Pl/Sql block of code that lists the highest paid Instructor in each of the Department. It should make use of a function department_highest which returns the highest paid Instructor for the given branch.

```
create or replace function department_highest(d_name varchar)
return varchar as
instruc_name varchar(20);
begin
select name into instruc_name from instructor
natural join (select dept_name, max(salary) as max_sal from instructor group by dept_name)
where dept_name=d_name and salary=max_sal;
return instruc_name;
end;
/


declare
cursor c1 is select distinct(dept_name) from department;
begin
for d_name in c1 loop
dbms_output.put_line(department_highest(d_name.dept_name) || ', ' || d_name.dept_name);
end loop;
end;
/
```



```
Function created.


Statement processed.
Crick, Biology
Brandt, Comp. Sci.
Kim, Elec. Eng.
Wu, Finance
Califieri, History
Mozart, Music
Einstein, Physics
```

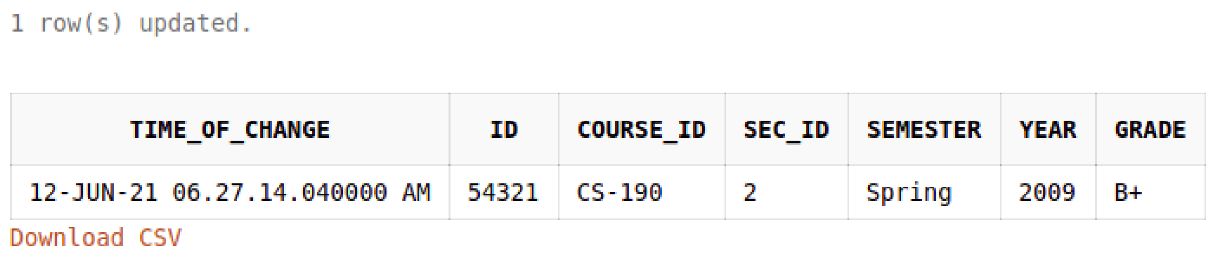
Row Triggers

1. Based on the University database Schema in Lab 2, write a row trigger that records along with the time any change made in the Takes (ID, course-id, sec-id, semester, year, grade) table in log_change_Takes (Time_Of_Change, ID, courseid,sec-id, semester, year, grade).

```
CREATE TABLE log_change_takes(
 time_of_change TIMESTAMP(2),
 ID varchar(5),
 course_id varchar(8),
 sec_id varchar(8),
 semester varchar(6),
 year number(4, 0),
 grade varchar(2)
);
CREATE OR REPLACE TRIGGER log_takes
AFTER UPDATE ON takes
FOR EACH ROW
BEGIN
 INSERT INTO log_change_takes VALUES(CURRENT_TIMESTAMP, :OLD.ID, :OLD.course_id,
:OLD.sec_id, :OLD.semester, :OLD.year, :OLD.grade);
END;

update takes set grade = 'A' where id = '54321' and course_id='CS-190' and sec_id='2' and
semester='Spring' and year='2009';

select * from log_change_takes;
```



```
1 row(s) updated.
```

TIME_OF_CHANGE	ID	COURSE_ID	SEC_ID	SEMESTER	YEAR	GRADE
12-JUN-21 06.27.14.040000 AM	54321	CS-190	2	Spring	2009	B+

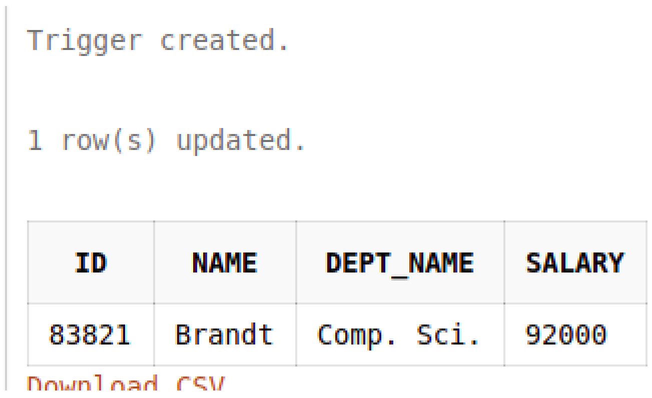
Download CSV

2. Based on the University database schema in Lab: 2, write a row trigger to insert the existing values of the Instructor (ID, name, dept-name, salary) table into a new table Old_ Data_Instructor (ID, name, dept-name, salary) when the salary table is updated.

```
CREATE TABLE old_data_instructor(
 ID varchar(5),
 name varchar(20),
 dept_name varchar(20),
 salary numeric(8, 2), check (salary > 29000)
);

CREATE OR REPLACE TRIGGER sal_trigger
AFTER UPDATE ON instructor
FOR EACH ROW
BEGIN
 INSERT INTO old_data_instructor VALUES(:OLD.ID, :OLD.name, :OLD.dept_name,
:OLD.salary);
END;
/

update instructor set salary = '95000' where id='83821';
select * from old_data_instructor;
```



```
Trigger created.

1 row(s) updated.
```

ID	NAME	DEPT_NAME	SALARY
83821	Brandt	Comp. Sci.	92000

Download CSV

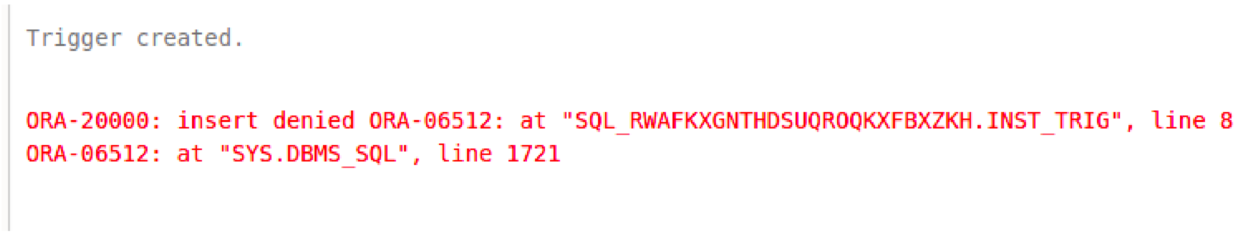
Database Triggers
3. Based on the University Schema, write a database trigger on Instructor that checks the following:
The name of the instructor is a valid name containing only alphabets.
The salary of an instructor is not zero and is positive
The salary does not exceed the budget of the department to which the instructor
Belongs.

```
create or replace trigger inst_trig
before insert on instructor--can't specify selected columns since insert is for a complete row
for each row

declare
        bud number(10);
begin
        select budget into bud from department where dept_name=:new.dept_name;

        if :new.name like '%0%' or :new.name like '%1%' or :new.name like '%2%' or :new.name
like '%3%' or :new.name like '%4%'
        or :new.name like '%5%' or :new.name like '%6%' or :new.name like '%7%' or :new.name
like '%8%' or :new.name like '%9%' then
                RAISE_APPLICATION_ERROR(-20000,'insert denied');
        end if;
        if :new.salary<=0 or :new.salary>bud then
                RAISE_APPLICATION_ERROR(-20000,'insert denied');
        end if;
end; -- declare keyword is specifically in capitals
/

insert into instructor values('10101', 'Srini123vasan', 'Comp. Sci.', '65000');
```



```
Trigger created.


ORA-20000: insert denied ORA-06512: at "SQL_RWAFKXGNTHDSUQROQKXFBXZKH.INST_TRIG", line 8
ORA-06512: at "SYS.DBMS_SQL", line 1721
```

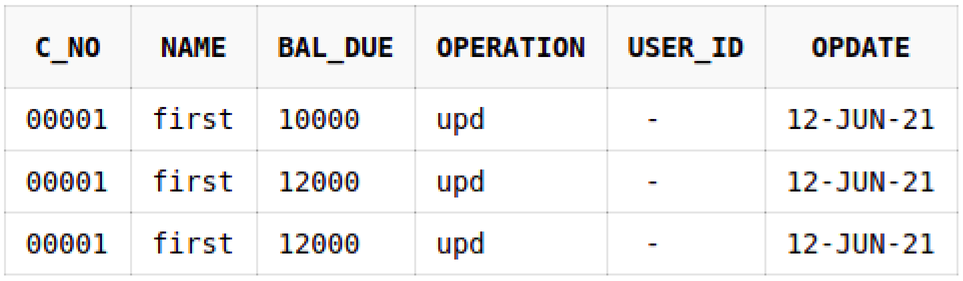
4. Create a transparent audit system for a table Client_master (client_no, name, address, Bal_due). The system must keep track of the records that are being deleted or updated. The functionality being when a record is deleted or modified the original record details and the date of operation are stored in the auditclient (client_no, name, bal_due, operation, userid, opdate) table, then the delete or update is allowed to go through.

```
CREATE table client_table
(c_no varchar(5) primary key,
name varchar(20),
bal_due number);

insert into client_table values ('00001','first',10000);
insert into client_table values ('00002','second',20000);
insert into client_table values ('00003','third',30000);

CREATE table audit_client
(c_no varchar(5),
name varchar(20),
bal_due number,
operation varchar(3),
user_id varchar(5) default('00000'),
opDate date);

CREATE or REPLACE trigger client_audit
BEFORE UPDATE or INSERT on client_table
FOR EACH ROW
BEGIN
CASE
WHEN UPDATING THEN
        insert into audit_client values
(:OLD.c_no,:OLD.name,:OLD.bal_due,'upd',NULL,sysdate);
WHEN DELETING THEN
        insert into audit_client values (:OLD.c_no,:OLD.name,:OLD.bal_due,'del',NULL,sysdate);
END CASE;
END;
/
```



C_NO	NAME	BAL_DUE	OPERATION	USER_ID	OPDATE
00001	first	10000	upd	-	12-JUN-21
00001	first	12000	upd	-	12-JUN-21
00001	first	12000	upd	-	12-JUN-21

Instead of Triggers
5. Based on the University database Schema in Lab 2, create a view Advisor_Student which is a natural join on Advisor, Student and Instructor tables. Create an INSTEAD OF trigger on Advisor_Student to enable the user to delete the corresponding entries in Advisor table.

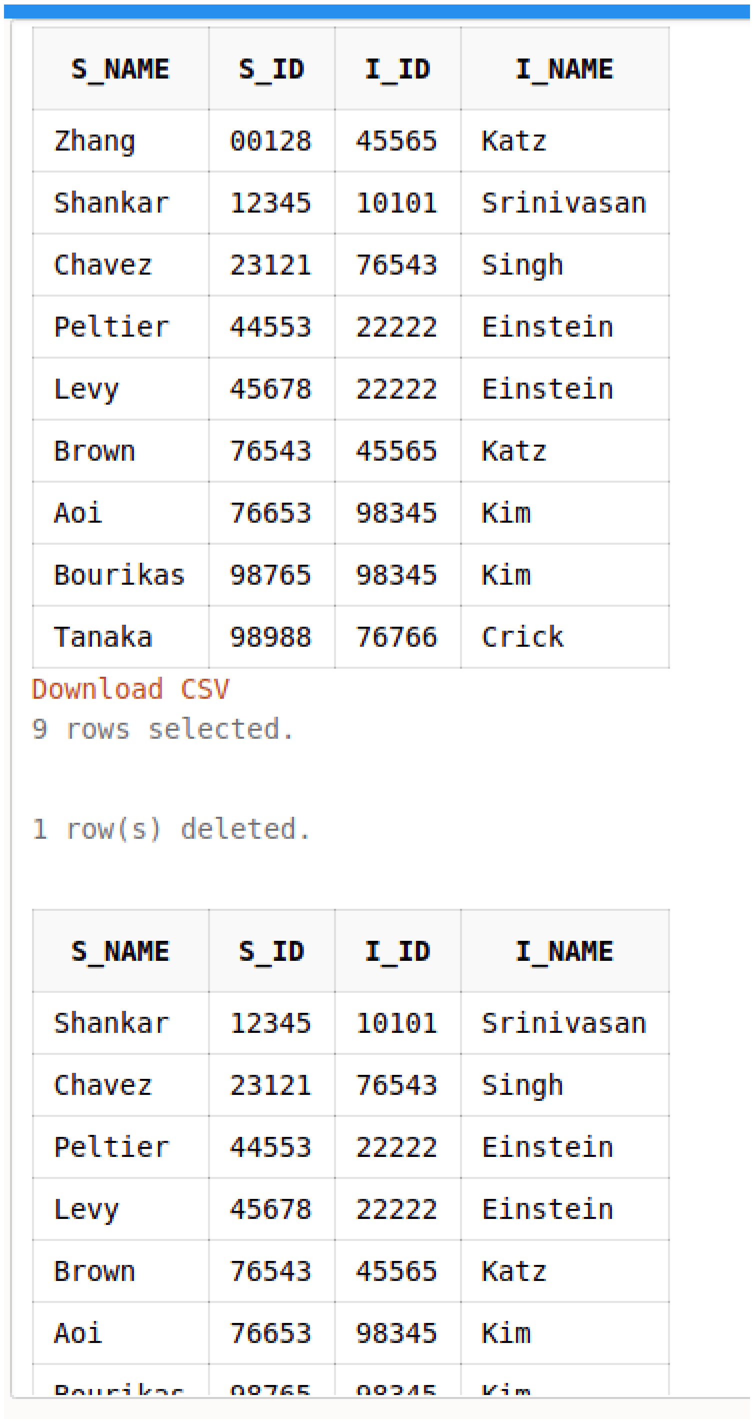
```
create view Advisor_Student as select s.name s_name, a.S_ID, a.I_ID, i.name i_name from
student s, advisor a, instructor i
where a.S_ID = s.ID and a.I_ID = i.ID;

create or replace trigger advisor_trigger
instead of delete on Advisor_Student
for each row
begin
delete from advisor where advisor.S_ID = :old.S_ID;
end;
/

select * from advisor_student;

delete from Advisor_Student where S_ID = '00128';

select * from advisor_student;
```



S_NAME	S_ID	I_ID	I_NAME
Zhang	00128	45565	Katz
Shankar	12345	10101	Srinivasan
Chavez	23121	76543	Singh
Peltier	44553	22222	Einstein
Levy	45678	22222	Einstein
Brown	76543	45565	Katz
Aoi	76653	98345	Kim
Bourikas	98765	98345	Kim
Tanaka	98988	76766	Crick

Download CSV

```
9 rows selected.

1 row(s) deleted.
```

S_NAME	S_ID	I_ID	I_NAME
Shankar	12345	10101	Srinivasan
Chavez	23121	76543	Singh
Peltier	44553	22222	Einstein
Levy	45678	22222	Einstein
Brown	76543	45565	Katz
Aoi	76653	98345	Kim
Bourikas	98765	98345	Kim