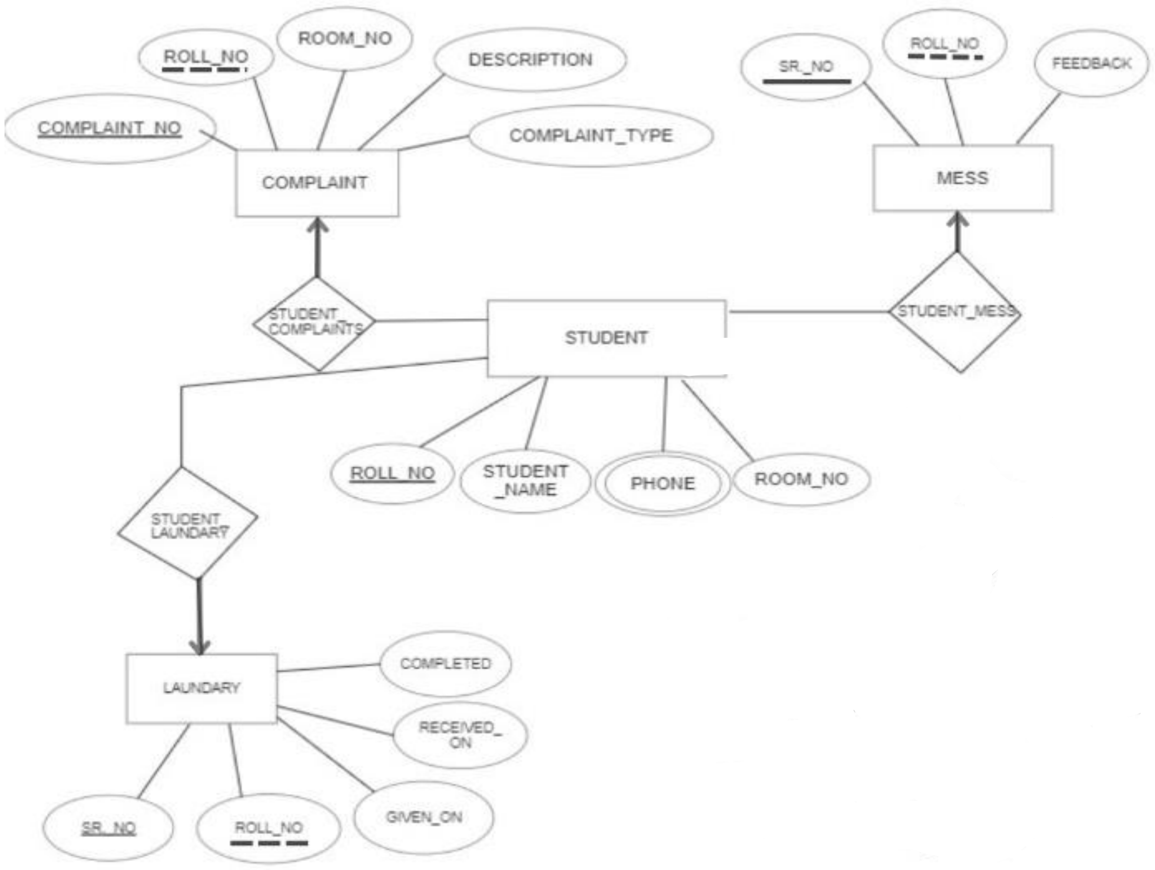
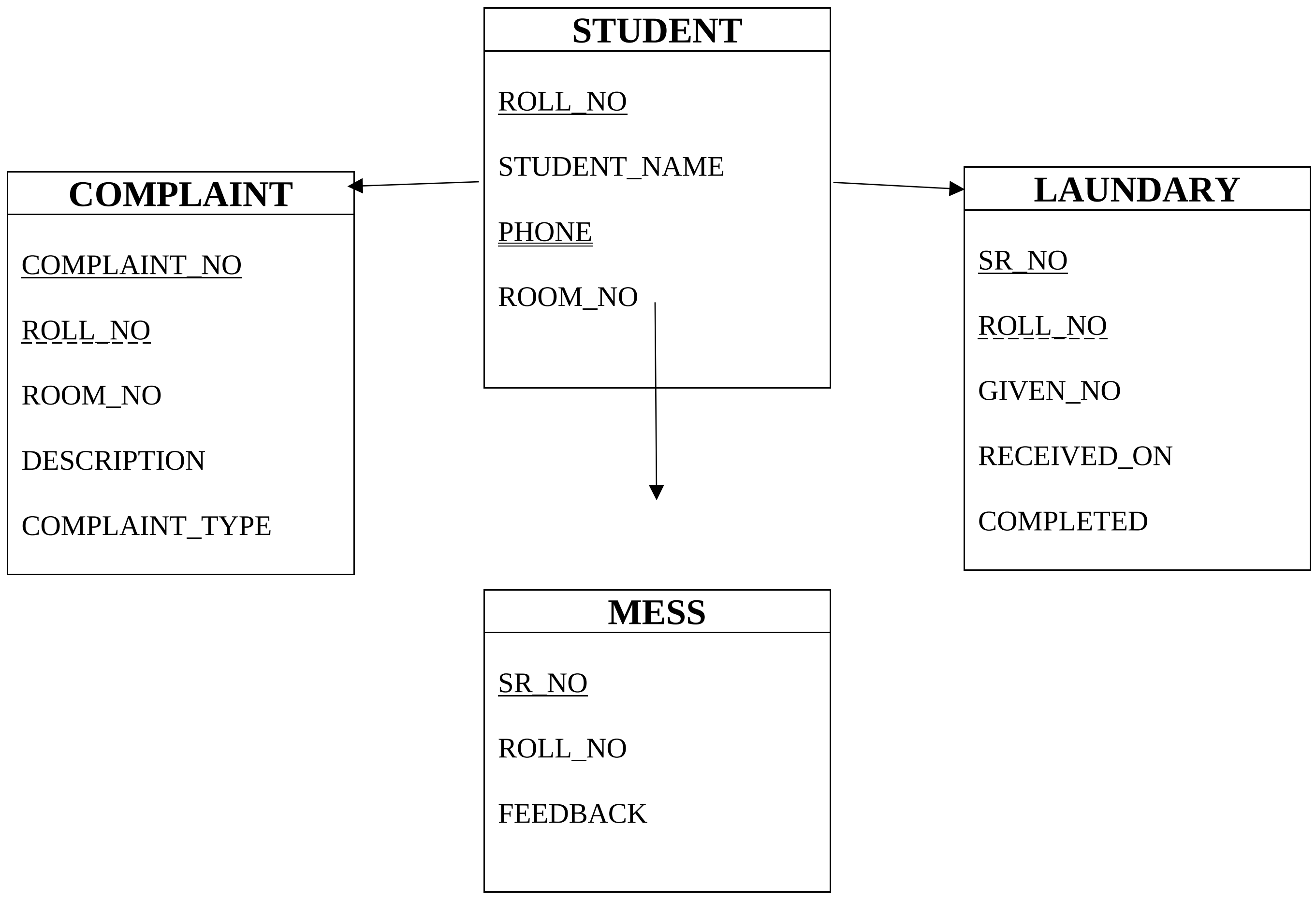
ER- DIAGRAM


ROLL_NO
ROOM_NO
DESCRIPTION
COMPLAINT_NO
COMPLAINT_TYPE
COMPLAINT
SR_NO
ROLL_NO
FEEDBACK
MESS
STUDENT_COMPLAINTS
STUDENT
STUDENT_MESS
ROLL_NO
STUDENT_NAME
PHONE
ROOM_NO
STUDENT LAUNDARY
LAUNDARY
COMPLETED
RECEIVED_ON
SR_NO
ROLL_NO
GIVEN_ON

ER TO TABLE


STUDENT
ROLL_NO
STUDENT_NAME
PHONE
ROOM_NO
COMPLAINT
COMPLAINT_NO
ROLL_NO
ROOM_NO
DESCRIPTION
COMPLAINT_TYPE
LAUNDARY
SR_NO
ROLL_NO
GIVEN_NO
RECEIVED_ON
COMPLETED
MESS
SR_NO
ROLL_NO
FEEDBACK

AIM:

Our project is based on Hostel Management. In our project, we have tried to modernize the conventional file-based registries still being used.

Description:

In this project, we have focused on 3 main departments namely Complaint Department, Mess Department, and the laundry department. In this project, we have used technologies like SQL and PL/SQL for various operations that can be performed in our database.

Normalization Process:

1NF- First Normal Form

If a relation contains a composite or multi-valued attribute, it violates the first normal form, or the relationship is in the first normal form if it does not contain any composite or multi-valued attribute. A relation is in its first normal form if every attribute in that relation is singled valued attribute.
A table is in 1 NF iff:

1. There are only Single Valued Attributes.
2. Attribute Domain does not change.
3. There is a unique name for every Attribute/Column.
4. The order in which data is stored does not matter.

Student Table

Roll No -- Roll No column satisfies all the above conditions.

Student_Name – Student_Name column satisfies all the above conditions.

Room_No – Room_no column satisfies all the above conditions.

Phone No – Here phone number is a multivalued column. To get our table in a 1NF form we need to make it a single-valued column. For that, we decompose the phone numbers into 2 different columns namely Phone_No1 and Phone_No2.

STUDENT

ROLL_NO	STUDENT_NAME	PHONE	ROOM_NO

ROLL_NO	STUDENT_NAME	PHONE_NO_1	PHONE_NO_2	ROOM_NO

Complaint Table

All the attributes satisfy the above 4 conditions. Our Complaint table is already in First Normal Form.

COMPLAINT

COMPLAINT_NO	ROLL_NO	ROOM_NO	DESCRIPTION	COMPLAINT_TYPE

Mess Table

All the attributes satisfy the above 4 conditions. Our Complaint table is already in First Normal Form.

MESS

SR_NO	ROLL_NO	FEEDBACK

Laundry Table

All the attributes satisfy the above 4 conditions. Our Complaint table is already in First Normal Form.

LAUNDRY

SR_NO	ROLL_NO	GIVEN_ON	RECEIVED_ON	COMPLETED

Now we have our database schema normalized to the First Normal Form.

2NF- Second Normal Form

To be in the second normal form, a relation must be in the first normal form and the relation must not contain any partial dependency. A relation is in 2NF if it has No Partial Dependency, i.e., no non-prime attribute (attributes that are not part of any candidate key) is dependent on any proper subset of any candidate key of the table.

Student Table

STUDENT

ROLL_NO	STUDENT_NAME

ROLL_NO	PHONE_NO_1	PHONE_NO_2

ROLL_NO	ROOM_NO

Complaint Table

COMPLAINT

COMPLAINT_NO	ROLL_NO

COMPLAINT_NO	DESCRIPTION	COMPLAINT_TYPE

Mess Table

MESS

SR_NO	ROLL_NO

SR_NO	FEEDBACK

Laundry Table

LAUNDARY

SR_NO	ROLL_NO

SR_NO	GIVEN_ON	RECEIVED_ON	COMPLETED

3NF- Third Normal Form

A relation that is in First and Second Normal Form and in which no non-primary-key attribute is transitively dependent on the primary key, then it is in Third Normal Form (3NF). If A->B and B->C are two FDs then A->C is called transitive dependency.

Student Table

ROLL_NO	STUDENT_NAME

ROLL_NO	PHONE_NO_1	PHONE_NO_2

ROLL_NO	ROOM_NO

Complaint Table

COMPLAINT_NO	ROLL_NO

COMPLAINT_NO	DESCRIPTION	COMPLAINT_TYPE

Mess Table

SR_NO	ROLL_NO

SR_NO	FEEDBACK

Laundry Table

SR_NO	ROLL_NO

SR_NO	GIVEN_ON	RECEIVED_ON	COMPLETED

BCNF

BCNF is the advanced version of 3NF. It is stricter than 3NF. A table is in BCNF if every functional dependency X → Y, X is the super key of the table. For BCNF, the table should be in 3NF, and for every FD, LHS is super key.

Student Table

ROLL_NO	STUDENT_NAME

ROLL_NO	PHONE_NO_1	PHONE_NO_2

ROLL_NO	ROOM_NO

Complaint Table

COMPLAINT_NO	ROLL_NO

COMPLAINT_NO	DESCRIPTION	COMPLAINT_TYPE

Mess Table

SR_NO	ROLL_NO

SR_NO	FEEDBACK

Laundry Table

SR_NO	ROLL_NO

SR_NO	GIVEN_ON	RECEIVED_ON	COMPLETED

4NF- Fourth Normal Form

The fourth normal form (4NF) is a level of database normalization where there are no non-trivial multivalued dependencies other than a candidate key. It builds on the first three normal forms (1NF, 2NF, and 3NF) and the Boyce-Codd Normal Form (BCNF). It states that, in addition to a database meeting the requirements of BCNF, it must not contain more than one multivalued dependency.

Properties – A relation R is in 4NF if and only if the following conditions are satisfied:

1. It should be in the Boyce-Codd Normal Form (BCNF).
2. the table should not have any Multi-valued Dependency.

Student Table

ROLL_NO	STUDENT_NAME

ROLL_NO	PHONE_NO_1

ROLL_NO	PHONE_NO_2

ROLL_NO	ROOM_NO

Complaint Table

COMPLAINT_NO	ROLL_NO

COMPLAINT_NO	DESCRIPTION	COMPLAINT_TYPE

Mess Table

SR_NO	ROLL_NO

SR_NO	FEEDBACK

Laundry Table

SR_NO	ROLL_NO

SR_NO	GIVEN_ON	RECEIVED_ON	COMPLETED

5NF- Fifth Normal Form

A relation R is in 5NF if and only if every join dependency in R is implied by the candidate keys of R. A relation decomposed into two relations must have loss-less join Property, which ensures that no spurious or extra tuples are generated when relations are reunited through a natural join.

Properties – A relation R is in 5NF if and only if it satisfies the following conditions:

1. R should be already in 4NF.
2. It cannot be further no loss decomposed (join dependency)

Student Table

ROLL_NO	STUDENT_NAME

ROLL_NO	PHONE_NO_1

ROLL_NO	PHONE_NO_2

ROLL_NO	ROOM_NO

Complaint Table

COMPLAINT_NO	ROLL_NO

COMPLAINT_NO	DESCRIPTION	COMPLAINT_TYPE

Mess Table

SR_NO	ROLL_NO

SR_NO	FEEDBACK

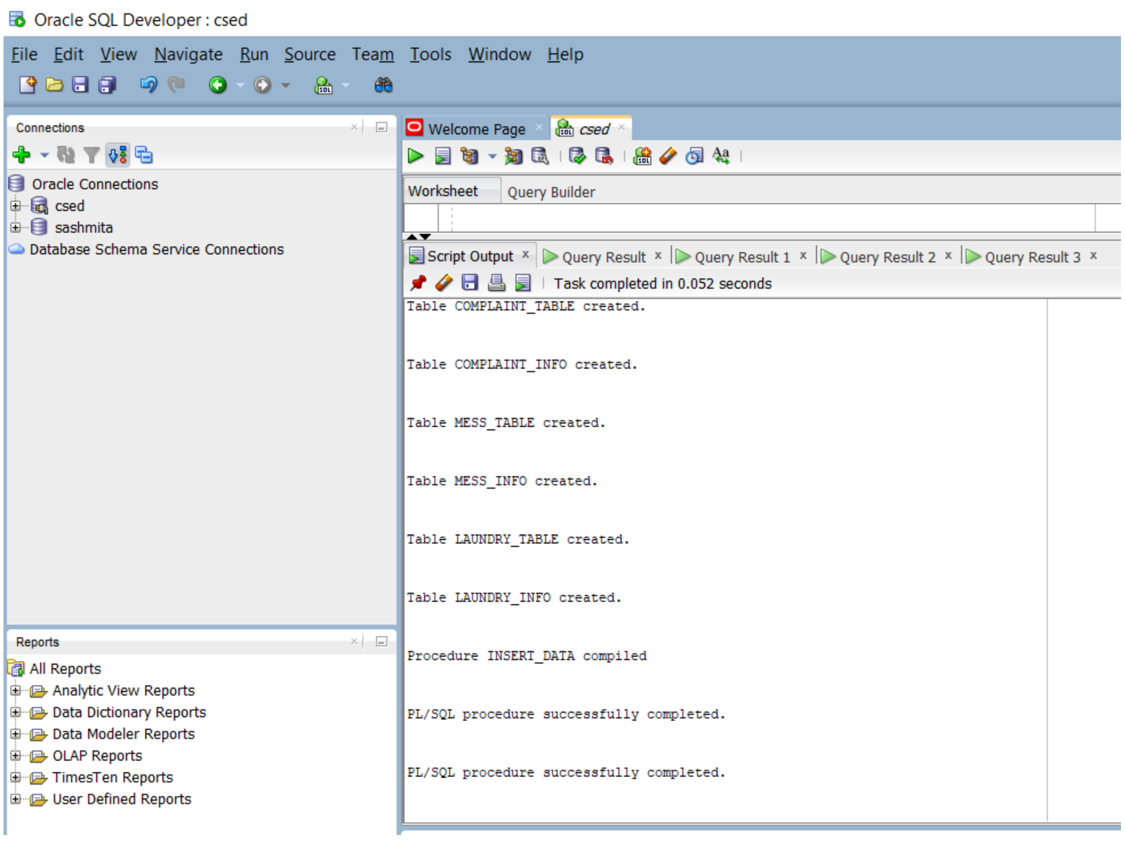
Laundry Table

SR_NO	ROLL_NO

SR_NO	GIVEN_ON	RECEIVED_ON	COMPLETED

SQL COMMANDS TO CREATE TABLE

```
create table student_n(
roll_no number(20) primary key ,
student_name varchar2(20)
);

create table student_ph1(
roll_no number(20) primary key  references student_n(roll_no),
student_phone1 number(10) );


create  table student_ph2(
roll_no number(20) primary key  references student_n(roll_no),
student_phone2 number(10) );

create table student_r(
roll_no number(20) primary key  references student_n(roll_no),
student_room_no number(5));


create table complaint_table(
complaint_no number(10) primary key,roll_no number(20) references student_n(roll_no));

create table complaint_info(complaint_no number(10) primary key references
complaint_table(complaint_no),description varchar2(100),
complaint_type varchar2(20));

create table mess_table(
sr_no number(10) primary key,
roll_no number(20) references student_n(roll_no));


create table mess_info(
sr_no number(10) primary key references mess_table(sr_no),
feedback varchar2(100));


create table laundry_table(
sr_no number(10) primary key,
roll_no number(20) references student_n(roll_no));


create table laundry_info(
sr_no number(10) primary key references laundry_table(sr_no),
given_on date,recieved_on date ,completed varchar2(1));
```



```
Table COMPLAINT_TABLE created.


Table COMPLAINT_INFO created.


Table MESS_TABLE created.


Table MESS_INFO created.


Table LAUNDRY_TABLE created.


Table LAUNDRY_INFO created.


Procedure INSERT_DATA compiled


PL/SQL procedure successfully completed.


PL/SQL procedure successfully completed.
```

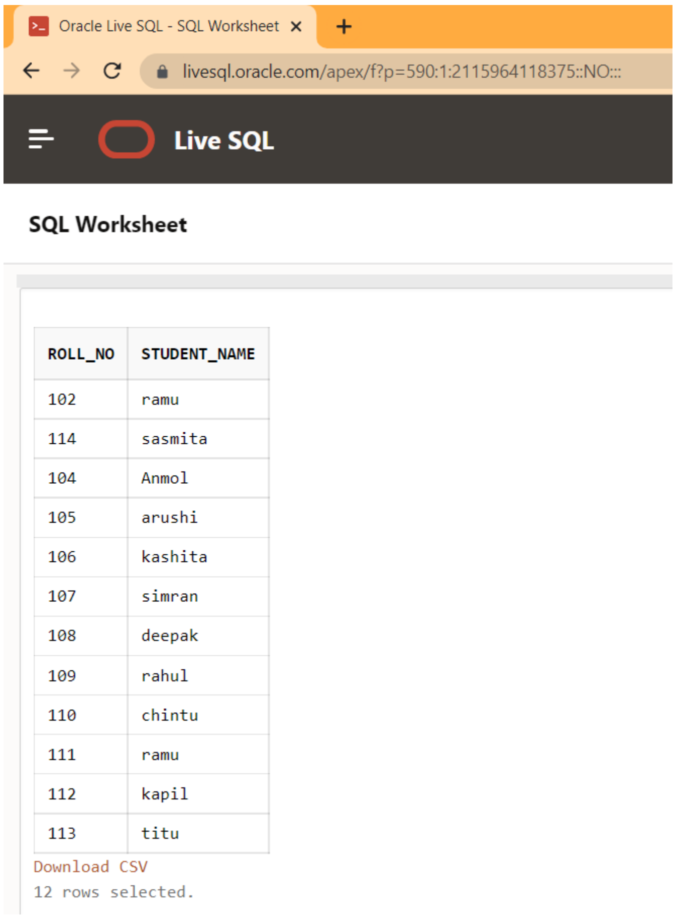
PL/SQL COMMANDS FOR INSERTION:-

For student table:-

```
CREATE OR REPLACE PROCEDURE insert_data (
roll student_n.roll_no%TYPE,
name student_n.student_name%TYPE,
phone1 student_ph1.student_phone1%TYPE,
phone2 student_ph2.student_phone2%TYPE,
room student_r.student_room_no%TYPE)
IS
BEGIN
INSERT INTO student_n (roll_no, student_name)
VALUES (roll,name);
INSERT INTO student_ph1 (roll_no, student_phone1)
VALUES (roll,phone1);
INSERT INTO student_ph2 (roll_no, student_phone2)
VALUES (roll,phone2);
INSERT INTO student_r(roll_no, student_room_no)
VALUES (roll,room);
COMMIT;
END;
/
```

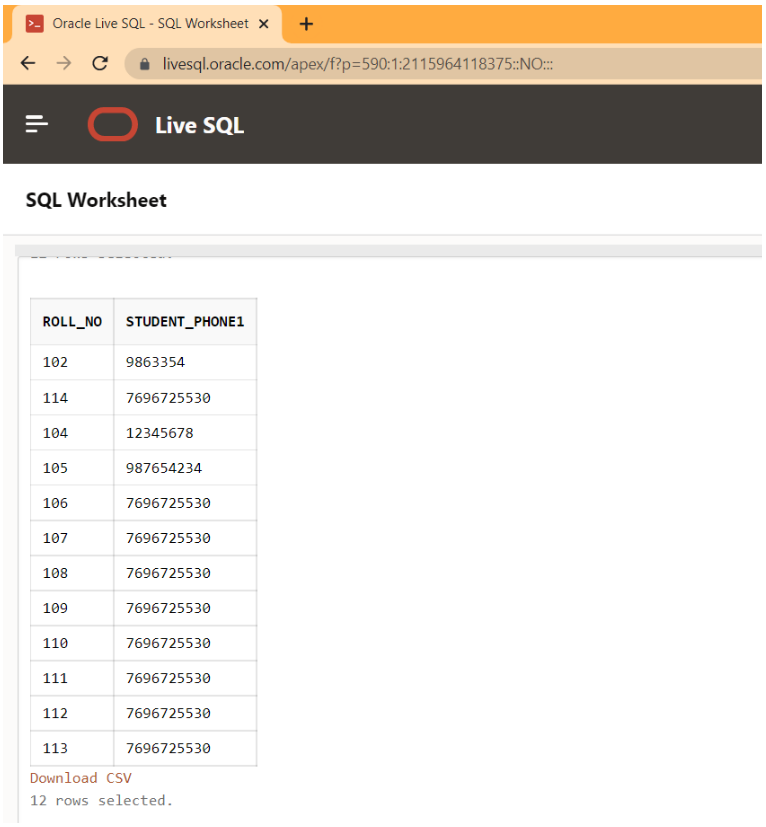
```
begin
insert_data(102,'ramu',9863354,47534724,13);
insert_data(114,'sasmita',7696725530,12345678,20);
insert_data(104,'Anmol',12345678,8765432455,21);
insert_data(105,'arushi',987654234,98756637,22);
insert_data(106,'kashita',7696725530,4374623473,23);
insert_data(107,'simran',7696725530,12345678,24);
insert_data(108,'deepak',7696725530,12345678,25);
insert_data(109,'rahul',7696725530,12345678,26);
insert_data(110,'chintu',7696725530,12345678,27);
insert_data(111,'ramu',7696725530,12345678,28);
insert_data(112,'kapil',7696725530,12345678,29);
insert_data(113,'titu',7696725530,12345678,30);
end;
/
```

```
select * from student_n;
select * from student_ph1;
select * from student_ph2;
select * from student_r;
```


Oracle Live SQL - SQL Worksheet
livesql.oracle.com/apex/f?p=590:1:2115964118375::NO:::
Live SQL
SQL Worksheet

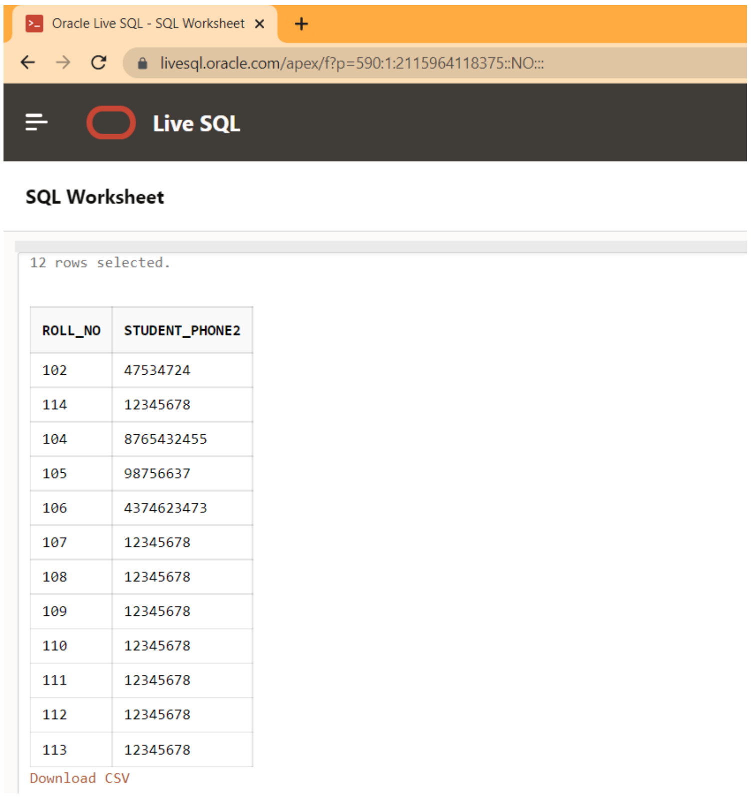
ROLL_NO	STUDENT_NAME
102	ramu
114	sasmita
104	Anmol
105	arushi
106	kashita
107	simran
108	deepak
109	rahul
110	chintu
111	ramu
112	kapil
113	titu

Download CSV
12 rows selected.


Oracle Live SQL - SQL Worksheet
livesql.oracle.com/apex/f?p=590:1:2115964118375::NO:::
Live SQL
SQL Worksheet

ROLL_NO	STUDENT_PHONE1
102	9863354
114	7696725530
104	12345678
105	987654234
106	7696725530
107	7696725530
108	7696725530
109	7696725530
110	7696725530
111	7696725530
112	7696725530
113	7696725530

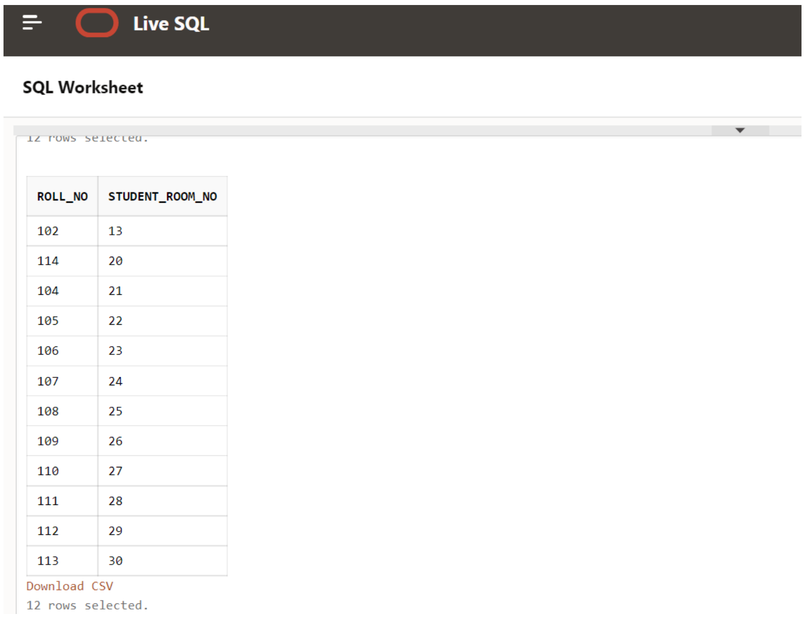
```
Download CSV
12 rows selected.
```


Oracle Live SQL - SQL Worksheet
livesql.oracle.com/apex/f?p=590:1:2115964118375::NO:::
Live SQL
SQL Worksheet

```
12 rows selected.
```

ROLL_NO	STUDENT_PHONE2
102	47534724
114	12345678
104	8765432455
105	98756637
106	4374623473
107	12345678
108	12345678
109	12345678
110	12345678
111	12345678
112	12345678
113	12345678

```
Download CSV
```



12 rows selected.

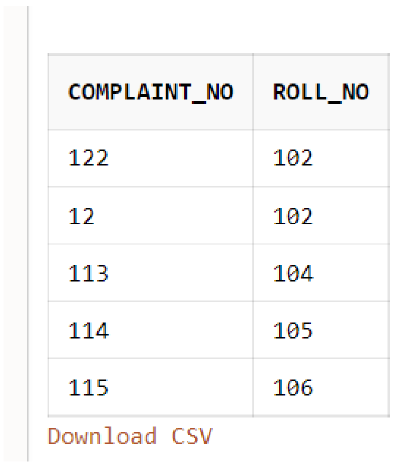
ROLL_NO	STUDENT_ROOM_NO
102	13
114	20
104	21
105	22
106	23
107	24
108	25
109	26
110	27
111	28
112	29
113	30

Download CSV
12 rows selected.

For complaint table:-

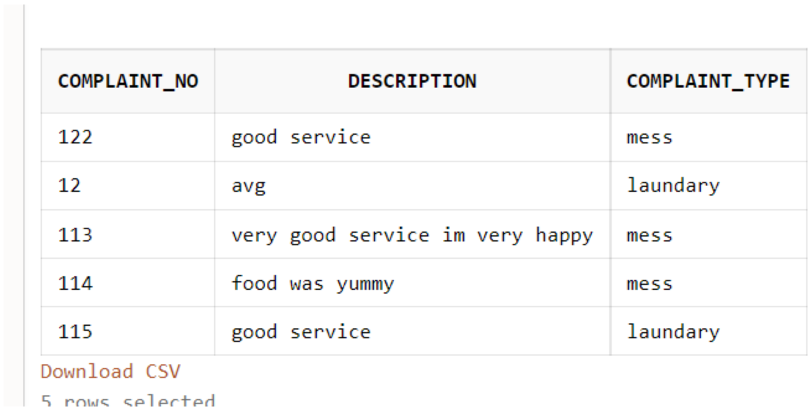
```
create or replace procedure add_complaint(
c_no complaint_table.complaint_no%type,
roll complaint_table.roll_no%type,
disc complaint_info.description%type,
c_type complaint_info.complaint_type%type
)
is
begin
insert into complaint_table(complaint_no,roll_no)
values(c_no,roll);
insert into complaint_info(complaint_no,description,complaint_type)
values(c_no,disc,c_type);
commit;
end;
/
```

```
begin
add_complaint(122,102,'good service','mess');
add_complaint(12,102,'avg','laundary');
add_complaint(113,104,'very good service im very happy','mess');
add_complaint(114,105,'food was yummy','mess');
add_complaint(115,106,'good service','laundary');
end;

select * from complaint_table;
select * from complaint_info;
```



COMPLAINT_NO	ROLL_NO
122	102
12	102
113	104
114	105
115	106

Download CSV



COMPLAINT_NO	DESCRIPTION	COMPLAINT_TYPE
122	good service	mess
12	avg	laundary
113	very good service im very happy	mess
114	food was yummy	mess
115	good service	laundary

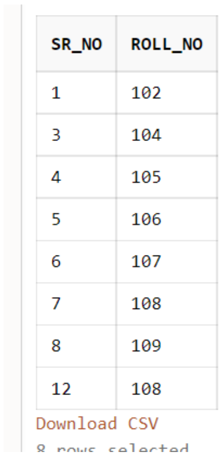
Download CSV

5 rows selected

For mess table:-

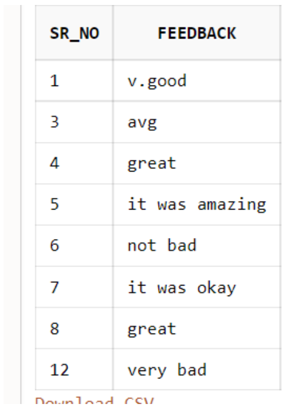
```
create or replace procedure add_mess(
sno mess_table.sr_no%type,
roll mess_table.roll_no%type,
feed mess_info.feedback%type
)
is
begin
insert into mess_table(sr_no,roll_no)
values(sno,roll);
insert into mess_info(sr_no,feedback)
values(sno,feed);
commit;
end;/
```

```
declare
sno mess_table.sr_no%type;
begin
select max(sr_no)into sno from mess_table;
sno:=sno+1;
add_mess(sno,102,'v.v.v.good');
add_mess(sno,102,'v.good');
add_mess(sno,108,'very bad');
add_mess(sno,104,'avg');
add_mess(sno,105,'great');
add_mess(sno,106,'it was amazing');
add_mess(sno,107,'not bad');
add_mess(sno,108,'it was okay');
add_mess(sno,109,'great');
end;
/

select * from mess_table;
select * from mess_info;
```



SR_NO	ROLL_NO
1	102
3	104
4	105
5	106
6	107
7	108
8	109
12	108

Download CSV
8 rows selected



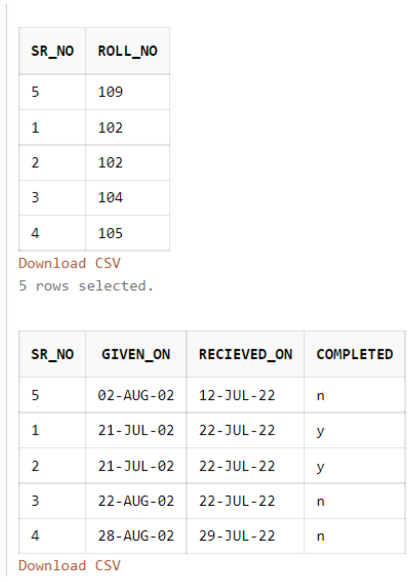
SR_NO	FEEDBACK
1	v.good
3	avg
4	great
5	it was amazing
6	not bad
7	it was okay
8	great
12	very bad

Download CSV

For laundry table:-

```
create or replace procedure add_laundry(
sno laundry_table.sr_no%type,
roll laundry_table.roll_no%type,
g_date laundry_info.given_on%type,
r_date laundry_info.recieved_on%type,
comp laundry_info.completed%type
)
is
begin
insert into laundry_table(sr_no,roll_no)
values(sno,roll);
insert into laundry_info(sr_no,given_on,recieved_on,completed)
values(sno,g_date,r_date,comp);
commit;
end;
/


declare
sno laundry_table.sr_no%type;
begin
select max(sr_no) into sno from laundry_table;
sno:=sno+1;
add_laundry(sno,109,to_date('2-08-2002','dd-mm-yyyy'),to_date('12-07-2022','dd-mm-yyyy'),'n');

end;
```

```
select * from laundry_table;
select * from laundry_info;
```



SR_NO	ROLL_NO
5	109
1	102
2	102
3	104
4	105

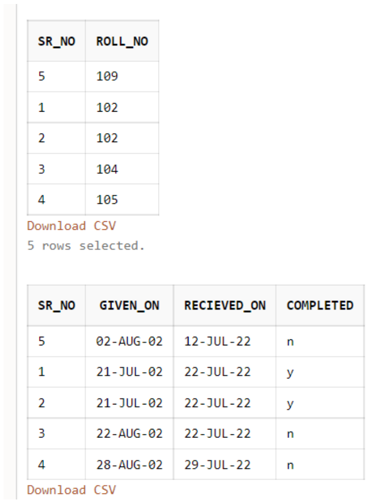
Download CSV
5 rows selected.

SR_NO	GIVEN_ON	RECIEVED_ON	COMPLETED
5	02-AUG-02	12-JUL-22	n
1	21-JUL-02	22-JUL-22	y
2	21-JUL-02	22-JUL-22	y
3	22-AUG-02	22-JUL-22	n
4	28-AUG-02	29-JUL-22	n

Download CSV

For update:-

```
create or replace procedure update_laundry(
sno laundry_table.sr_no%type,
comp laundry_info.completed%type
)
is
begin
update laundry_info set completed = comp where sr_no = sno;
commit;
end;
/


begin
update_laundry(5,'y');
end;


select * from laundry_info;
```



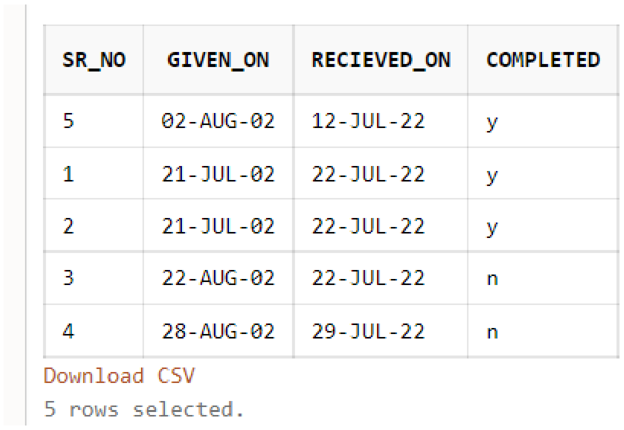
SR_NO	ROLL_NO
5	109
1	102
2	102
3	104
4	105

Download CSV
5 rows selected.

SR_NO	GIVEN_ON	RECIEVED_ON	COMPLETED
5	02-AUG-02	12-JUL-22	n
1	21-JUL-02	22-JUL-22	y
2	21-JUL-02	22-JUL-22	y
3	22-AUG-02	22-JUL-22	n
4	28-AUG-02	29-JUL-22	n

Download CSV

After Updation:-



SR_NO	GIVEN_ON	RECIEVED_ON	COMPLETED
5	02-AUG-02	12-JUL-22	y
1	21-JUL-02	22-JUL-22	y
2	21-JUL-02	22-JUL-22	y
3	22-AUG-02	22-JUL-22	n
4	28-AUG-02	29-JUL-22	n

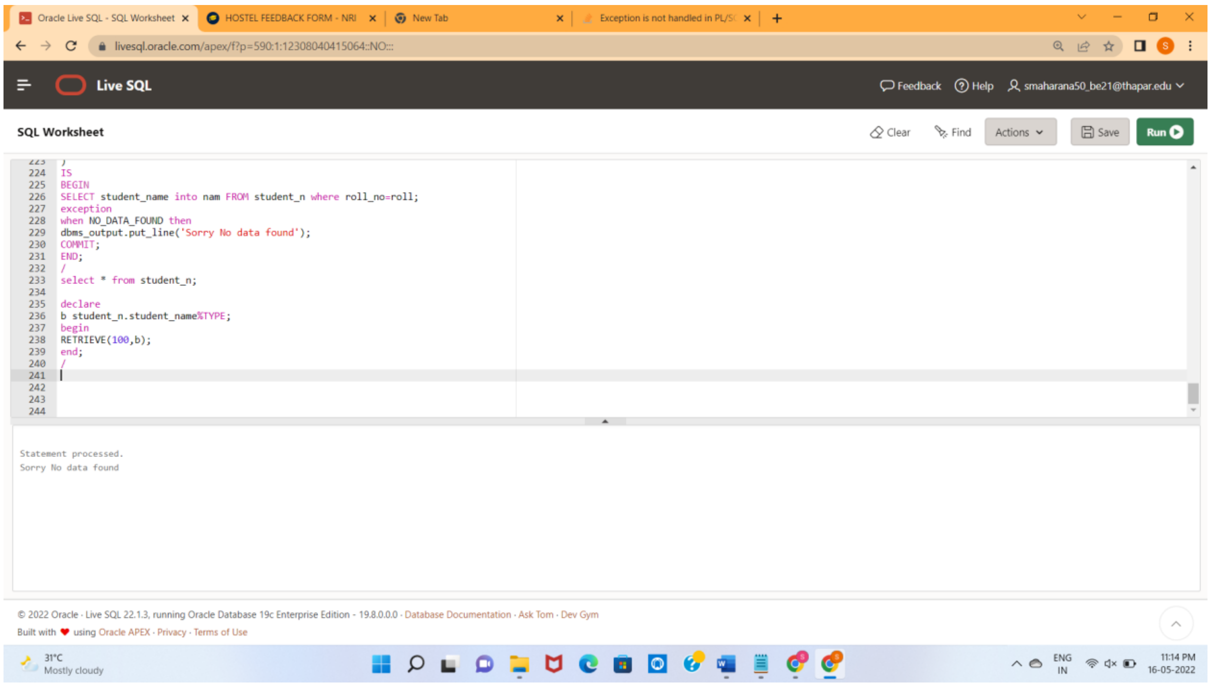
Download CSV
5 rows selected.

For trigger:-

```
create or replace trigger Insert_at_12
before insert
on student_n
for each row
when ((to_char(sysdate,'fmDAY'))=('MONDAY'))
declare
abcd exception;
begin
raise abcd;
exception
when abcd then
dbms_output.put_line('have a good start of the week.');
end;
/



insert into student_n values(1200,'anmol');
select * from student_n;
select to_char(sysdate,'day') from dual;
```

Exception Included Procedure:-

```
CREATE OR REPLACE PROCEDURE RETRIEVE(
roll student_n.roll_no%TYPE,
nam OUT student_n.student_name%TYPE
)
IS
BEGIN
SELECT student_name into nam FROM student_n where roll_no=roll;
exception
when NO_DATA_FOUND then
dbms_output.put_line('Sorry No data found');
COMMIT;
END;
/

select * from student_n;

declare
b student_n.student_name%TYPE;
begin
RETRIEVE(100,b);
end;
```


Oracle Live SQL - SQL Worksheet
HOSTEL FEEDBACK FORM - NRI
New Tab
Exception is not handled in PL/S
livesql.oracle.com/apex/f?p=590:1:12308040415064::NO:::
Live SQL
Feedback
Help
smaharana50_be21@thapar.edu
SQL Worksheet
Clear
Find
Actions
Save
Run

```
)
IS
BEGIN
SELECT student_name into nam FROM student_n where roll_no=roll;
exception
when NO_DATA_FOUND then
dbms_output.put_line('Sorry No data found');
COMMIT;
END;
/
select * from student_n;

declare
b student_n.student_name%TYPE;
begin
RETRIEVE(100,b);
end;
/
```

```
Statement processed.
Sorry No data found
```

© 2022 Oracle · Live SQL 22.1.3, running Oracle Database 19c Enterprise Edition - 19.8.0.0.0 · Database Documentation · Ask Tom · Dev Gym
Built with ♥ using Oracle APEX · Privacy · Terms of Use