**THAPAR INSTITUTE OF ENGINEERING AND TECHNOLOGY**



**UCS310**

**Database Management System**

**Project Report On**

**HOSTEL MANAGEMENT SYSTEM**

Submitted By :

RIDDHI GARG 102103282

SUKHMANJOT KAUR 102103291

JAGDISH AGARWAL 102103266

JANARDHAN 102103269

**INDEX**

|  |  |
| --- | --- |
| **S.No.** | **Topic** |
| 1 | Introduction |
| 2 | ER Diagram |
| 3 | ER Diagram To Table |
| 4 | Functional Dependencies And Normalization |
| 5 | SQL Queries To Create Tables |
| 6 | Triggers |
| 7 | Functions |
| 8 | Cursors |
| 9 | Procedures |

**Introduction**

In this project, we have designed a database management system to organize and store information about college hostel management System. This database contains information about students, the rooms, hostels, laundry and messes they are assigned to. It also stores admin information and complaints that are lodged by the students. The main aim of this project is to efficiently store and retrieve student information. Our design helps facilitate convenient management of data, by computerizing most of the work and getting rid of manual entry and record systems.

**ER DIAGRAM**

Diagram

Description automatically generated

N 1

N 1

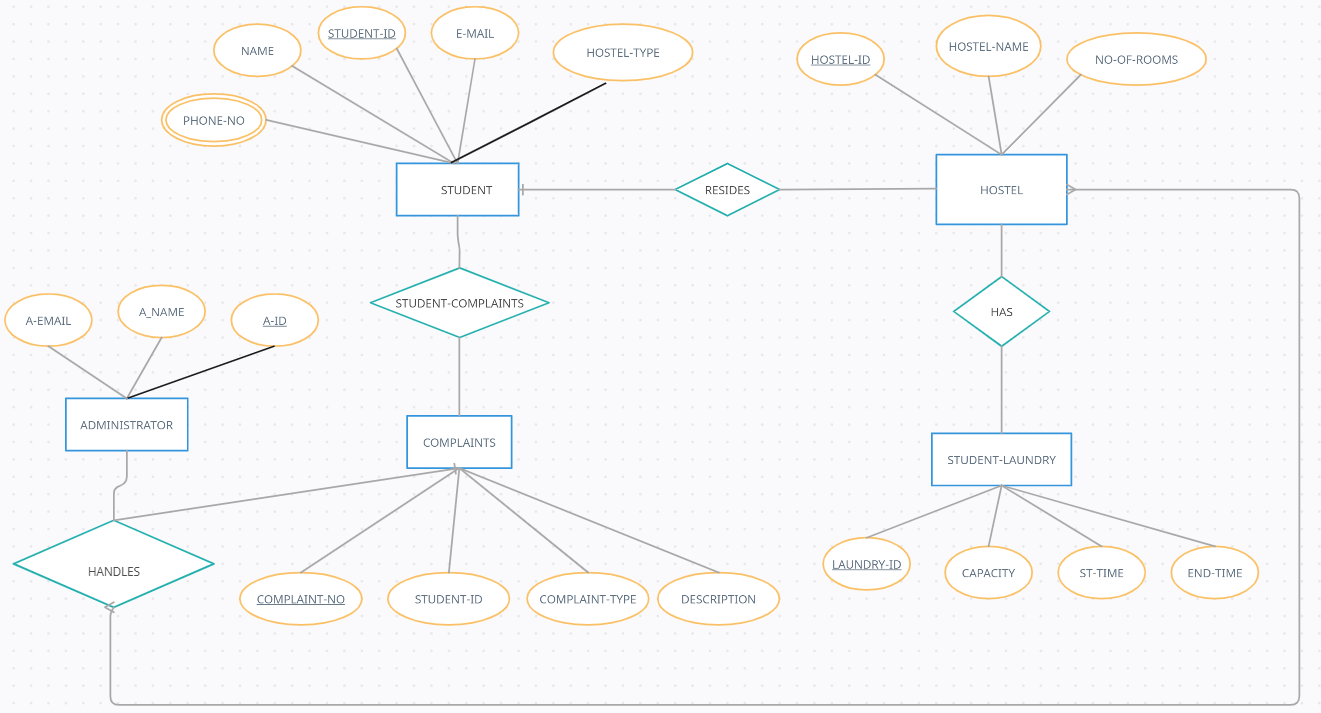
1

N

N

N

1



**ER-DIAGRAM TO TABLE**

Diagram

Description automatically generated Timeline

Description automatically generated

**FUNCTIONAL-DEPENDENCY**

**AND**

**NORMALISATION**

1. Student

**Functional Dependencies**:

STUDENT\_ID → NAME, EMAIL, HOSTEL-TYPE

On applying Attribute Closure , we get :

[STUDENT\_ID]+ = {NAME, EMAIL, PHONE, HOSTEL\_TYPE}

It can be seen that attribute Student-Id covers all the attributes of Student table . Hence,

**Primary Key**: Student-Id

**Normalisation :**

Student Table can be decompose into 2 tables:

1. Student (Student-Id , Name , EMail)
2. Hostel (Student-Id , Hostel-Type)

Now, Candidate Key in :

Student Table {Student-Id}

Hostel Table { Student-Id , Hostel-Type }

Both Tables are in 2NF and 3NF , as each table has only 1 theme and no partial dependencies or transitive dependencies.

1. Student-PhoneNo

**Functional Dependencies**:

SR-NO → STUDENT-ID , PHONE-NO

On applying Attribute Closure , we get :

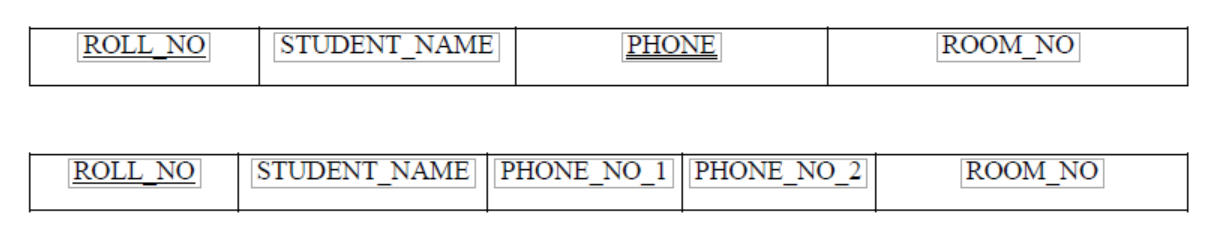
[SR-NO]+ = { STUDENT-ID , PHONE-NO }

It can be seen that attribute Sr-No covers all the attributes of Student-PhoneNo table . Hence,

**Primary Key**: Sr-No

**Normalisation :**

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.



1. Hostel

**Functional Dependencies:**

HOSTEL­­\_ID → HOSTEL\_NAME, NO\_OF\_ROOMS

On applying Attribute Closure , we get :

[HOSTEL\_ID]+ = { HOSTEL\_NAME, NO\_OF\_ROOMS}

It can be seen that attribute Hostel-Id covers all the attributes of Hostel table . Hence,

**Primary Key:** HOSTEL\_ID

**Normalisation :**

Hostel Table can be decompose into 2 tables:

1. Hostel (Hostel-Id , Hostel-Name)
2. HostelRooms (Hostel-Id , No-of-rooms)

Now, Candidate Key in :

Hostel Table {Hostel-Id}

HostelRooms Table { Hostel-Id }

Both Tables are in 3NF , as each table has only 1 theme and NO transitive dependencies.

1. Hostel-Mess

**Functional Dependencies:**

MESS\_NO → HOSTEL\_TYPE, MESS-EXPENDITURE , FEEDBACK

On applying Attribute Closure , we get :

[MESS\_NO]+ = { HOSTEL\_TYPE , MESS-EXPENDITURE , FEEDBACK}

It can be seen that attribute Mess-No covers all the attributes of Hostel-Mess table . Hence,

**Primary Key:** MESS\_NO

**Normalisation :**

The Table is already in 3NF since the table has no partial or transitive dependencies.

1. Administrator

**Functional Dependencies:**

A­­\_ID → A-NAME , A-EMAIL

On applying Attribute Closure , we get :

[A\_ID]+ = { A-NAME , A-EMAIL }

It can be seen that attribute A-Id covers all the attributes of Administrator table . Hence,

**Primary Key:** A\_ID

**Normalisation :**

Administrator Table can be decomposed into 2 tables:

1. Administrator (A-Id , A-Name , A-Email )
2. Hostel-Manager (A-Id , Hostel-Id)

Now, Candidate Key in :

Administrator Table {A-Id}

Hostel-Manager Table { A-Id , Hostel-Id }

Both Tables are in 2NF and 3NF , as each table has only 1 theme and no partial dependencies or transitive dependencies.

1. Student-Laundry

**Functional Dependencies:**

LAUNDRY­­\_ID → CAPACITY , ST-TIME , END-TIME

HOSTEL-ID → LAUNDRY\_ID

On applying Attribute Closure , we get :

[LAUNDRY\_ID]+ = { CAPACITY, ST-TIME, END-TIME}

It can be seen that attribute Laundry-Id covers all the attributes of Laundry table . Hence,

**Primary Key:** LAUNDRY\_ID

**Normalisation :**

Table can be decomposed into 3 tables:

1. Hostel (Hostel-Id)
2. Hostel-Laundry (Hostel-Id , Laundry-Id)
3. Laundry (Laundry-Id , Capacity , St-time , End-time )

Now, Candidate Key in :

Laundry Table {Laundry-Id}

Hostel Table { Hostel-Id }

Hostel-Laundry {Hostel-Id , Laundry-Id}

Both Tables are in 3NF , as each table has only 1 theme and no partial dependencies or transitive dependencies.

1. Student-Complaint

**Functional Dependencies :**

COMPLAINT\_NO → STUDENT\_ID

STUDENT\_ID → COMPLAINT\_TYPE, DESCRIPTION

On applying Attribute Closure , we get :

[COMPLAINT\_NO]+ = { STUDENT\_ID , COMPLAINT\_TYPE, DESCRIPTION }

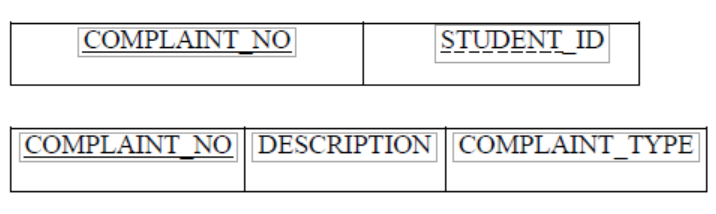
It can be seen that attribute Complaint-No covers all the attributes of Student-Complaint table . Hence,

**Primary Key:** Complaint-No

**Normalisation :**

Hostel Table can be decompose into 2 tables:

1. Complaint (Complaint-No , Complaint-Type , Complaint-Description)
2. Student (Student-Id)



Now, Candidate Key in :

Complaint Table {Complaint-No}

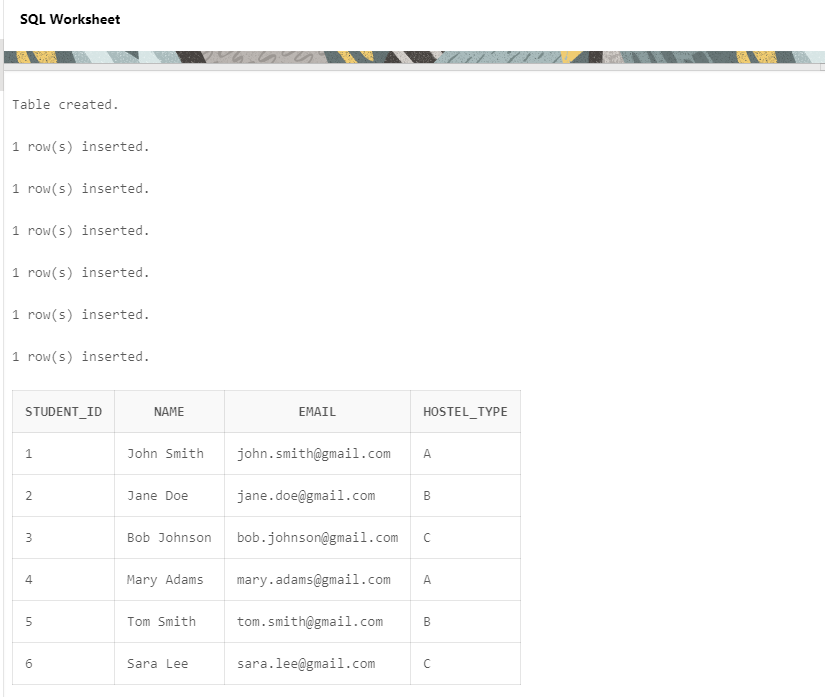
Student Table { Student-Id }

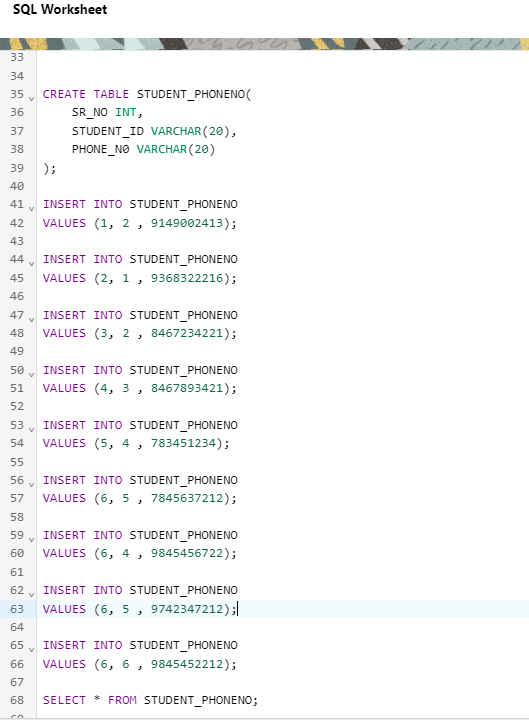
Both Tables are in 3NF , as each table has only 1 theme and no transitive dependencies.

**SQL QUERIES TO CREATE TABLE**

A picture containing text

Description automatically generated





Table

Description automatically generated

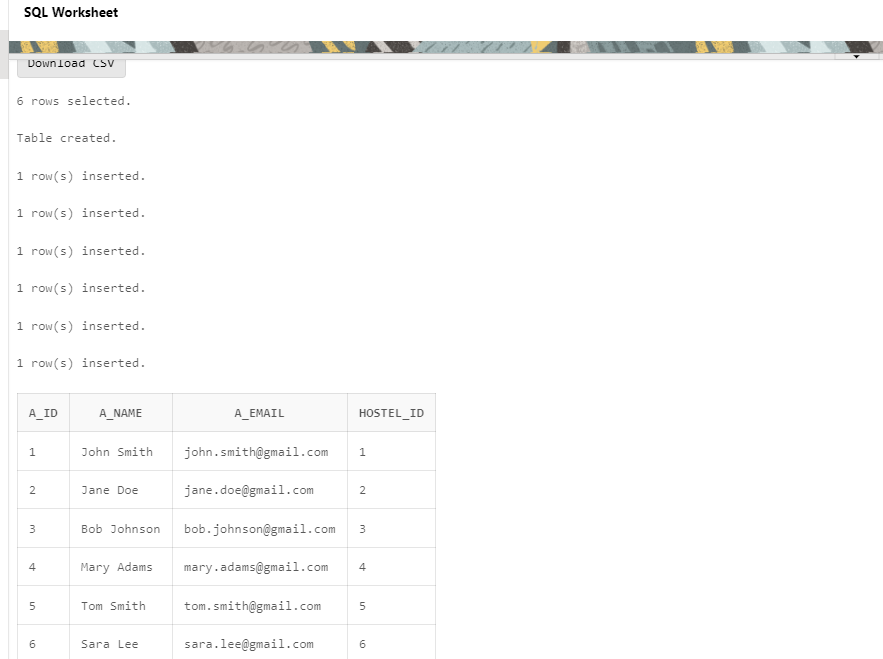
Text

Description automatically generated



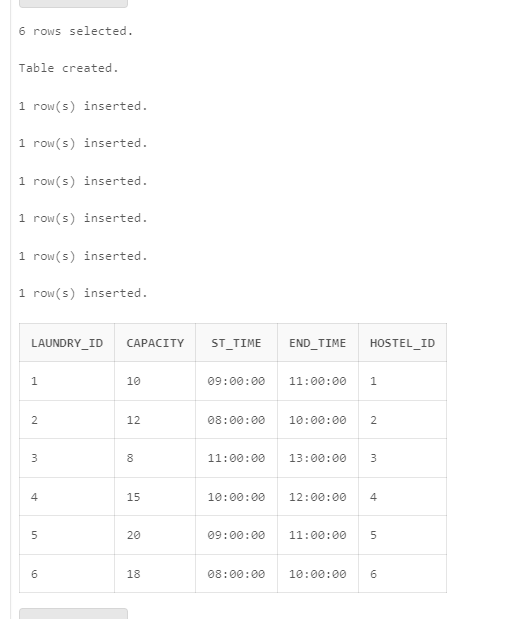
A picture containing text

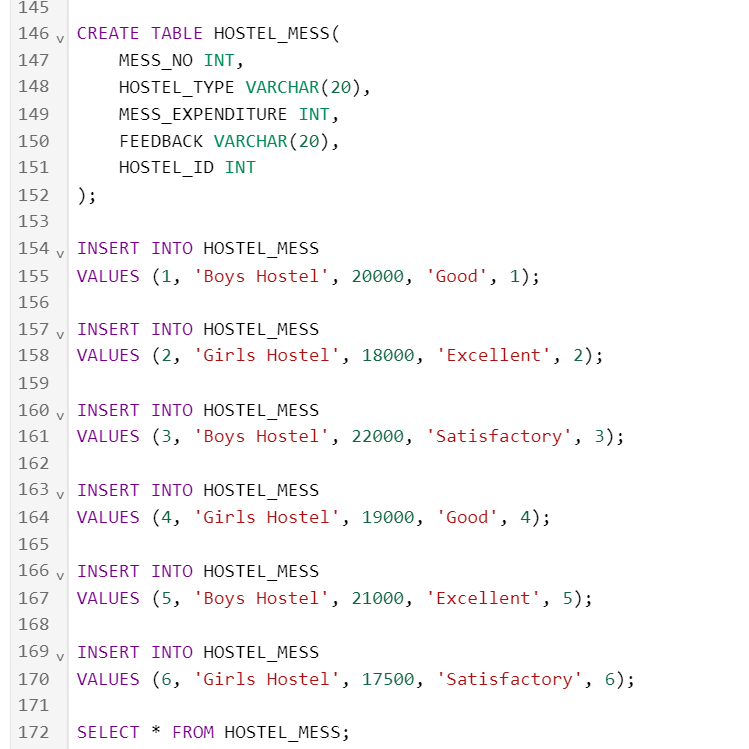
Description automatically generated



Text

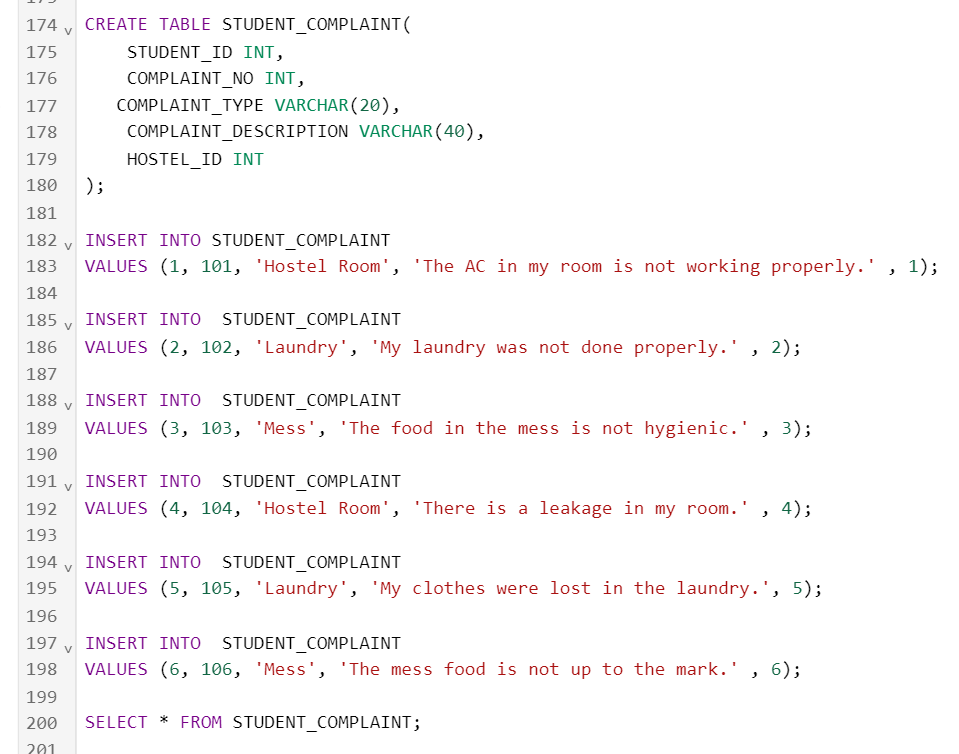
Description automatically generated with low confidence





Table

Description automatically generated



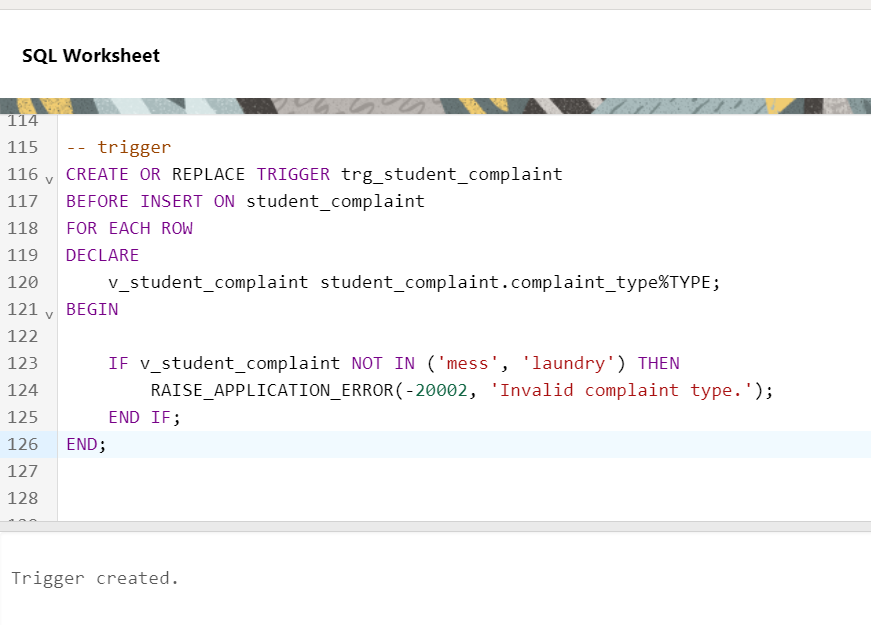
Table

Description automatically generated

**PL-SQL**

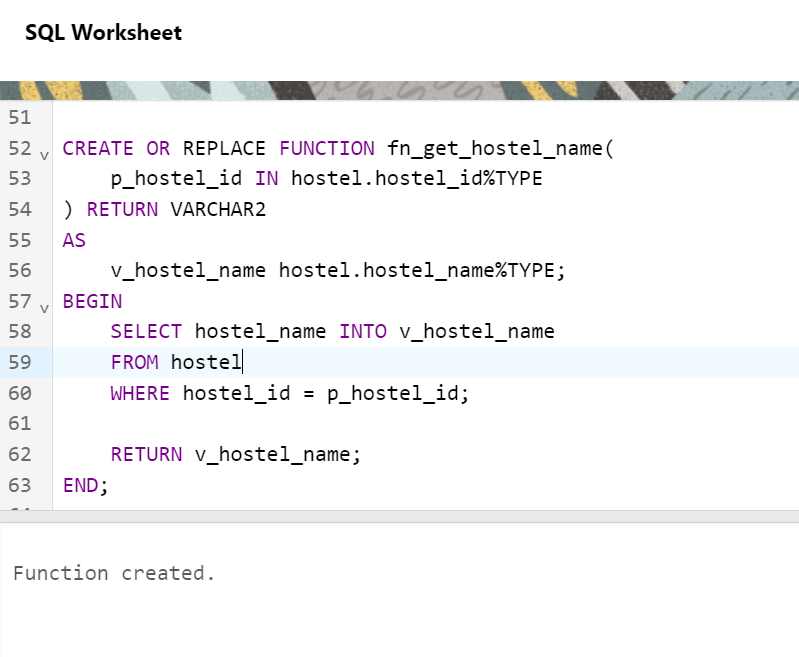
Triggers

CHECKS THAT COMPLAINT-TYPE IS MESS OR LAUNDRY



**FUNCTIONS**

RETURNS HOSTEL-NAME FROM HOSTEL-TABLE



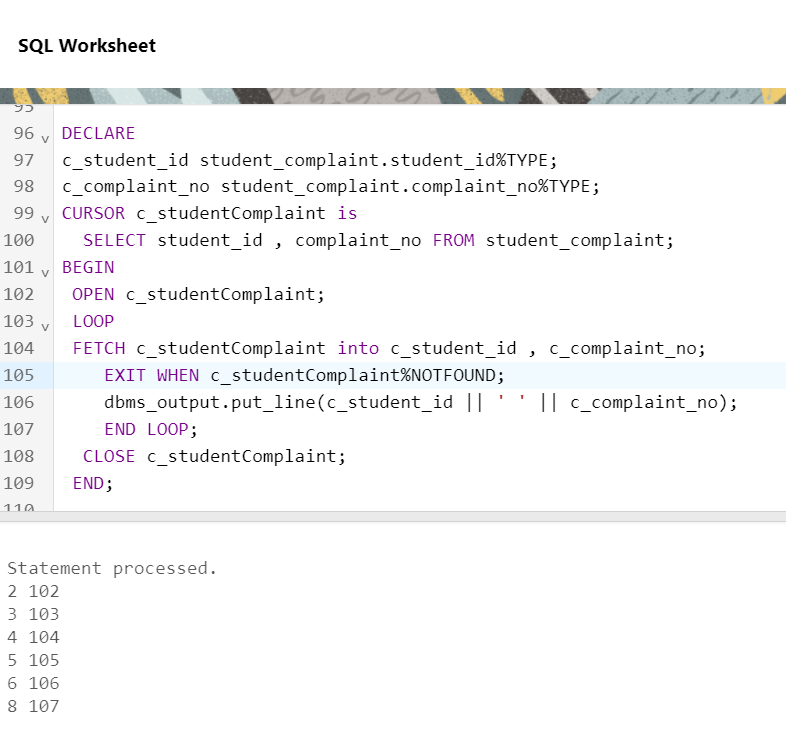
Function-output

Graphical user interface, text, application, email

Description automatically generated

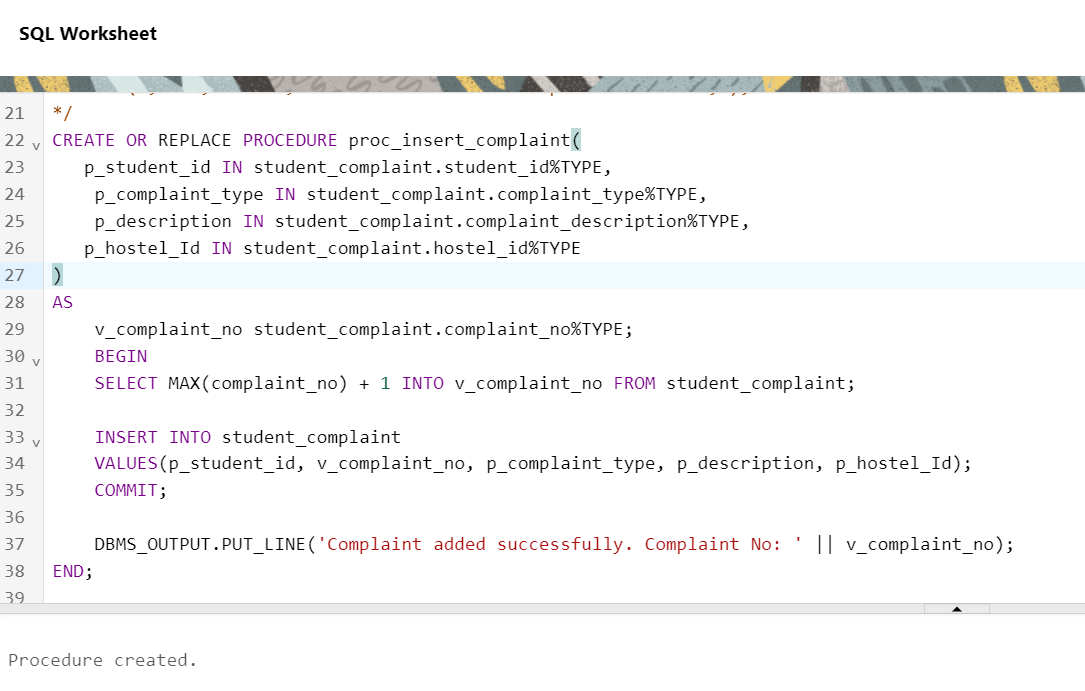
**CURSORS**

DISPLAYING STUDENT-ID AND COMPLAINT-NO FROM STUDENT-COMPLAINT-TABLE



**PROCEDURES**

INSERTING NEW COMPLAINT INTO STUDENT-COMPLAINT TABLE



output



**CONCLUSION**

In Conclusion , Our Project closely resembles the real-life-model of Hostel-Management-System . Moreover , the PL/SQL also have been created in such a way so as to help the hostel-System in their future.