Project team 14 - Ticketing Tool

TEAM MEMBERS:

LIKITHA CHOWDARY PUNATI
SAGAR SHARMA
PRATIK A KONTAMWAR
SRISANMATHI RAMACHANDRAN

PROJECT PROPOSAL

CONTEXT:
The goal is to create a software that keeps track of issues related to IT products used in the company.

OBJECTIVE:
We are going to design a database for the above application that maintains the record of the tickets raised by the organisation.

SCOPE:
The scope primarily consists of two types of users, Business users and IT contractor users. The users from the Business side consists of customer support representatives, employees and managers. The users from IT contractor side consist of customer support representatives, IT managers and engineers working on the ticket. There could be multiple level of hierarchy of engineers involved in a ticket. This defines the scope of our database project where every user can view,delete,update the database according to the business rules..

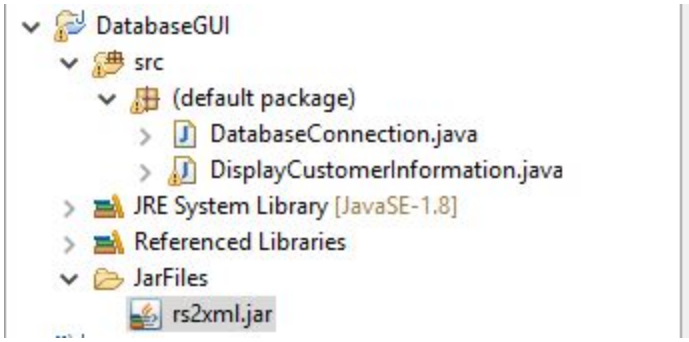
Project: Sprint 0 - Environment setup and high level requirements & conceptual design

Part 1: Environment setup

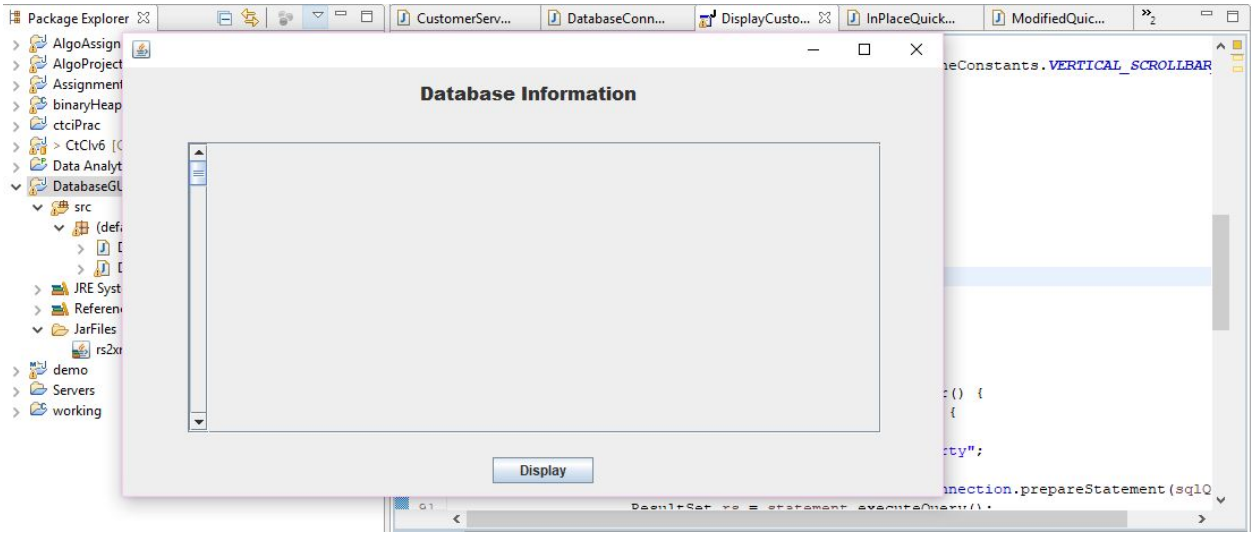
The Environment for displaying UI involves java swing and Jtable. The service runs on JDK 1.8 .

The UI will display a message “database connection successful” once the connection has been established. On clicking the display button, the information for the property table which is there in RDS instance at AWS, will be displayed.

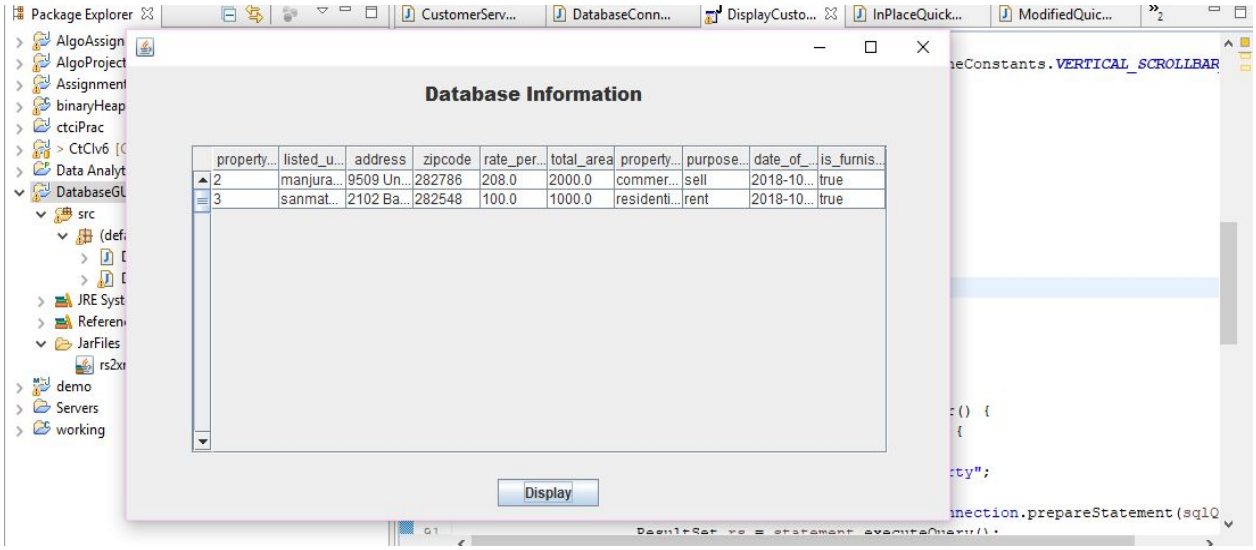
Here are the screenshot for the same:



```java
import java.sql.*;

public class DatabaseConnection {

    Connection dbConnection = null;

    public static Connection dataConnection() {
        Connection con = null;
        try {
            Class.forName("com.mysql.cj.jdbc.Driver");
             con = DriverManager.getConnection(
                    "jdbc:mysql://fall2018dbsagarsharma.cbtgolwyo970.us-east-2.rds.ama
            // here                                                  ername and admin password

             JOption                                                 abase connection is successful");


            //con.cl
        } catch (Exc
            JOptionPane.showMessageDialog(null, e);
            System.out.println(e);
        }

        return con;
    }
```



On clicking display, the information stored in the database is displayed



Part 2: High level requirements

Initial User Roles

User Roles	**Description**
Customer care executive of the business unit	This customer care executive is an employee of the Business unit who interacts with the end user
Manager of the business unit	This manager is the employee of the business unit who manages other employees
Engineer of the business unit	This engineer is the employee of the business unit who reports to the managers and solve issues related to the business unit
Customer care executive of the IT service provider	This customer care executive is an employee of the IT service provider who interacts with any of the employees of the business unit
Manager of the IT service provider	This manager is the employee of the IT service provider who manages other employees(namely Customer Care Executives and Engineer)

Engineer of the IT service provider	This engineer is the employee of the IT service provider who reports to the managers and solve issues related to the IT services raised by the business unit employees as well as employees within the organisation

Initial user story descriptions

User Stories	Description
US1	As an employee, I want to create/edit a ticket
US2	As an employee, I want to assign the ticket to another employee
US3	An employee logins/logout of account through which he/she will raise a ticket
US4	As a manager, I want to see the tickets assigned to a particular employee of my organisation
US5	As a manager, I want to see the tickets raised by particular employee of my organisation
US6	As a manager, I want to see the details of employee who have missed an SLA, so that I can access their performance
US7	As a manager, I want to search tickets based on date, SLA met/missed,topic
US8	As a manager, I want to search the employee/employees whose skillset help solve a particular issue

Part 3: High level conceptual design:

User entities:

Employee
Ticket
Service
Business Unit
Account
Department

Relationships:

Employee creates/edits Ticket
Employee login/logout of Account
Employee works for Business Unit
Employee works for Department**
Business Unit uses Service (of the IT company)
Employee assigns Ticket (to Employee)

** here department refers to the group of people that specialises in a particular skill set. For example cloud computing is a department

Sprint -1

Updated User Stories based on the refined Requirements
Reordering User Stories as well as modifying them we get :

User Stories	Description
US1	~~As an employee, I want to create/edit a ticket~~ As an employee of the IT contractor/Business Unit, I want to create a ticket in order to raise an issue.
US2	An employee logins/logout of account through which he/she will raise a ticket
US3	~~As a manager, I want to see the tickets assigned to a particular employee of my organisation~~ As a manager, I want to see the tickets assigned to my reportee of my department.
US4	As a manager, I want to see the tickets raised by particular employee of my organisation
US5	~~As a manager, I want to search the employee/employees whose skillset help solve a particular issue~~ As an Employee, I want to see the tickets assigned to me
US6	~~As an employee, I want to assign the ticket to another employee~~ As a employee of IT contractor/Business Unit, I want to assign the ticket to an employee after searching for relevant department concerned with the issue
US7	As a manager, I want to see the details of employee who have missed an SLA, so that I can access their performance

US8	As a manager, I want to search tickets based on date, SLA met/missed,topic

**The user stories highlighted will be considered for the current sprint.

Entity:

1)Employee

Attributes:

<u>emp_id</u>

name[composite]

first_name

last_name

phone_number[multi-valued]

email

designation

Primary Key justification: emp_id is taken as primary key since it is unique for every employee in the organisation. It is assumed that email id of an employee can change over time or with the transfer of employee to different location.

2)Ticket

Attributes:

<u>ticket_id</u>

ticket_type

end_user_email

end_user_phone_number

severity

priority

opened_date

closed_date

status

description

resolution

age[Derived]

**end_user_email and end_user_phone_number are related to the customer of the business user (if any) who is facing an issue and have raised an issue to the business unit

Primary Key Justification: ticket_id is an autoincrement primary key generated and assigned to a ticket at the time of its creation.

3)Account

Attributes:

<u>email_id</u>

password

Primary Key Justification: For every account of the employee, the employee will login with its email id

3)Department

Attributes:

<u>dept_id</u>

department_name

department_code

Primary Key Justification: dept_id is the autoincrement primary key used to keep track of the department.

**department have been built based on the domain of the business unit egs for business units of oil and gas domain, there would be a separate department for oil and natural gas business units in the IT company

4)BusinessUnit

Attributes:

<u>business_unit_id</u>

business_unit_name

Primary Key Justification: An auto increment primary key has been created to keep a track of the business units an IT company is working with

5)Assignment Group

Attributes:

<u>assignment_group_id</u>

assignment_group_name

Primary Key Justification: An auto increment primary key has been created to keep a track of groups in the IT organisation.

Relationships:

1) Employee login/logout of Account

Cardinality: one to one

Participation: Employee has total participation; Account has total participation

2) Employee creates Ticket

Cardinality : one to many

Participation: Employee has partial participation; Ticket has total participation

3)Employee has Ticket

Cardinality: one to many

Participation: Employee has partial participation; Ticket has total participation

4)Employee reports to Manager[Employee]

**this is a case of recursive relationship as manager is also an employee

Cardinality : many to one

Participation: Employee has total participation; Manager has partial participation

5)Department has Assignment Group

Cardinality: one to many

Participation:Department has total participation; Assignment group has total participation

6)Assignment Group has Employee

Cardinality: one to many

Participation: Assignment group has total participation; Employee has total participation

7)Business Unit has Employee

Cardinality:One to Many

Participation: Business Unit has total participation; Employee has total participation

8)Manager[Employee] manages the Department

Cardinality: one to one

Participation: Manager[Employee] has partial participation; Department has total participation

9)Team Lead[Employee] manages Assignment Group

Cardinality: one to one

Participation: Team Lead[Employee] has partial participation; Assignment Group has a total participation.

Logical Design

Table: Employee
Columns:

emp_id
first_name
last_name
manager_id[foreign key;references emp_id of the employee table]
phone_personal
phone_business
email_id
designation
password
business_unit_id[foreign key; references business_unit_id of the business unit table]

Justification: email_id, password,business_unit_id could have been the separate columns of the account table. But since it is total participation, we can include the columns in the employee table.

Table: Ticket
Columns:

ticket_id
ticket_type
end_user_email
end_user_phone_number
severity
priority
opened_date
closed_date
age[derived]
opened_by[foreign key; references emp_id of the employee table]
status
description
resolution
assigned_to[foreign key; references emp_id of the employee table]

Table: Department
Columns:

- department_id
- department_name
- department_code
- manager_id[foreign key;references emp_id of the employee table]

Table: BusinessUnit
Columns:

- business_unit_id
- business_unit_name

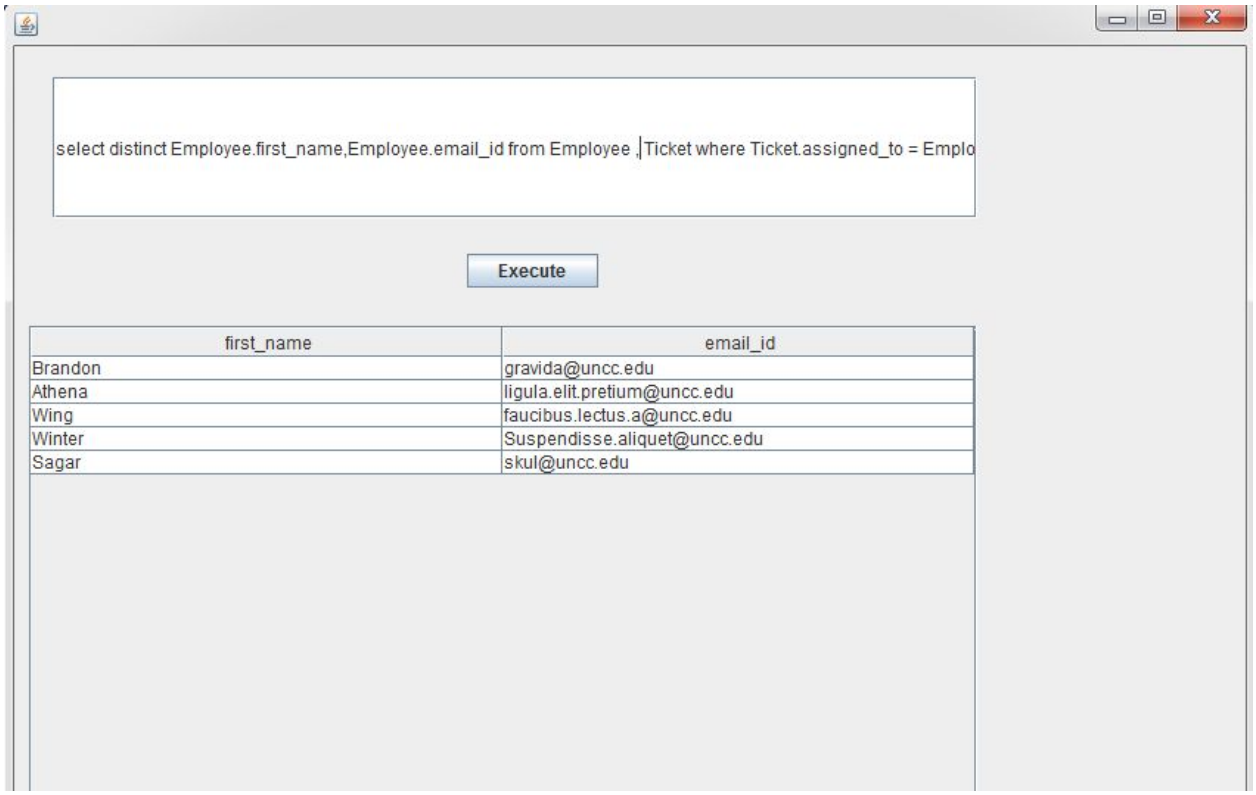
Table: AssignmentGroup
Columns:

- group_id
- group_name
- department_id[foreign key;references department_id from department]
- team_lead_id[foreign key;references emp_id from employee]

Sample Queries:

1)Get the names and email id of the employees distinctly who have tickets in open state

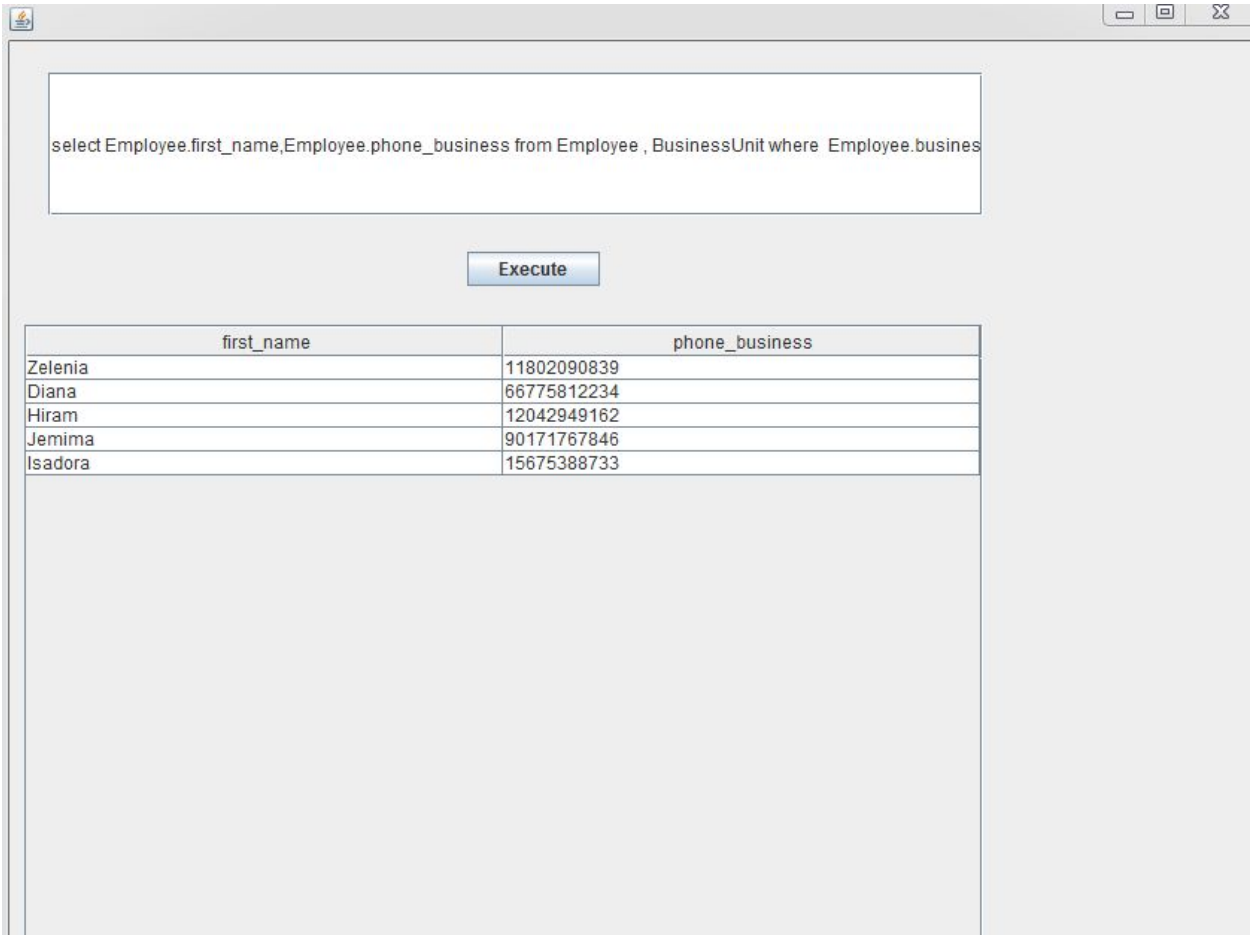
select distinct Employee.first_name,Employee.email_id from Employee , Ticket where Ticket.assigned_to = Employee.emp_id and Ticket.status = 'open';



first_name	email_id
Brandon	gravida@uncc.edu
Athena	ligula.elit.pretium@uncc.edu
Wing	faucibus.lectus.a@uncc.edu
Winter	Suspendisse.aliquet@uncc.edu
Sagar	skul@uncc.edu

2)Display the first name and business phone number of all the employee who doesnot belong to the UNCC IT TEAM.

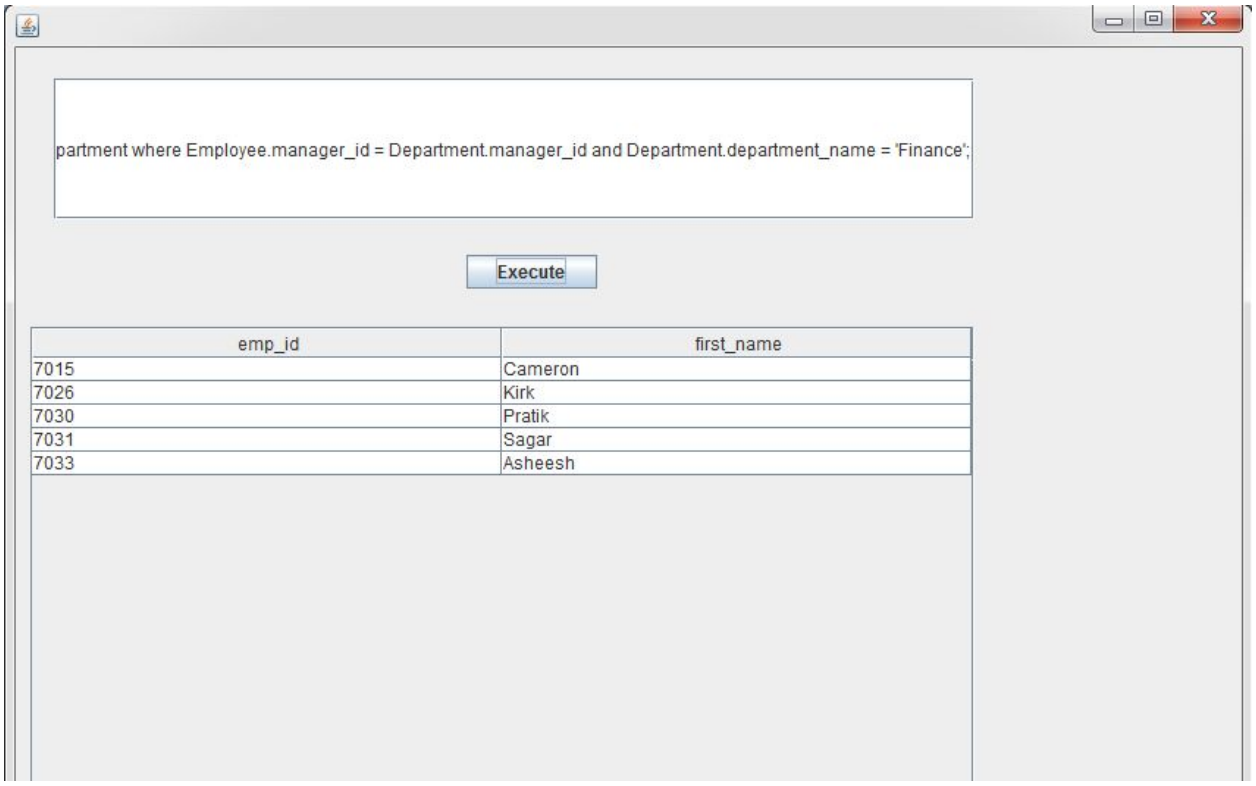
select Employee.first_name,Employee.phone_business from Employee , BusinessUnit where Employee.business_unit_id = BusinessUnit.business_unit_id and BusinessUnit.business_unit_name <> 'UNCC IT TEAM';



first_name	phone_business
Zelenia	11802090839
Diana	66775812234
Hiram	12042949162
Jemima	90171767846
Isadora	15675388733

3)Get the names of the Team leads who report to the manager of finance department

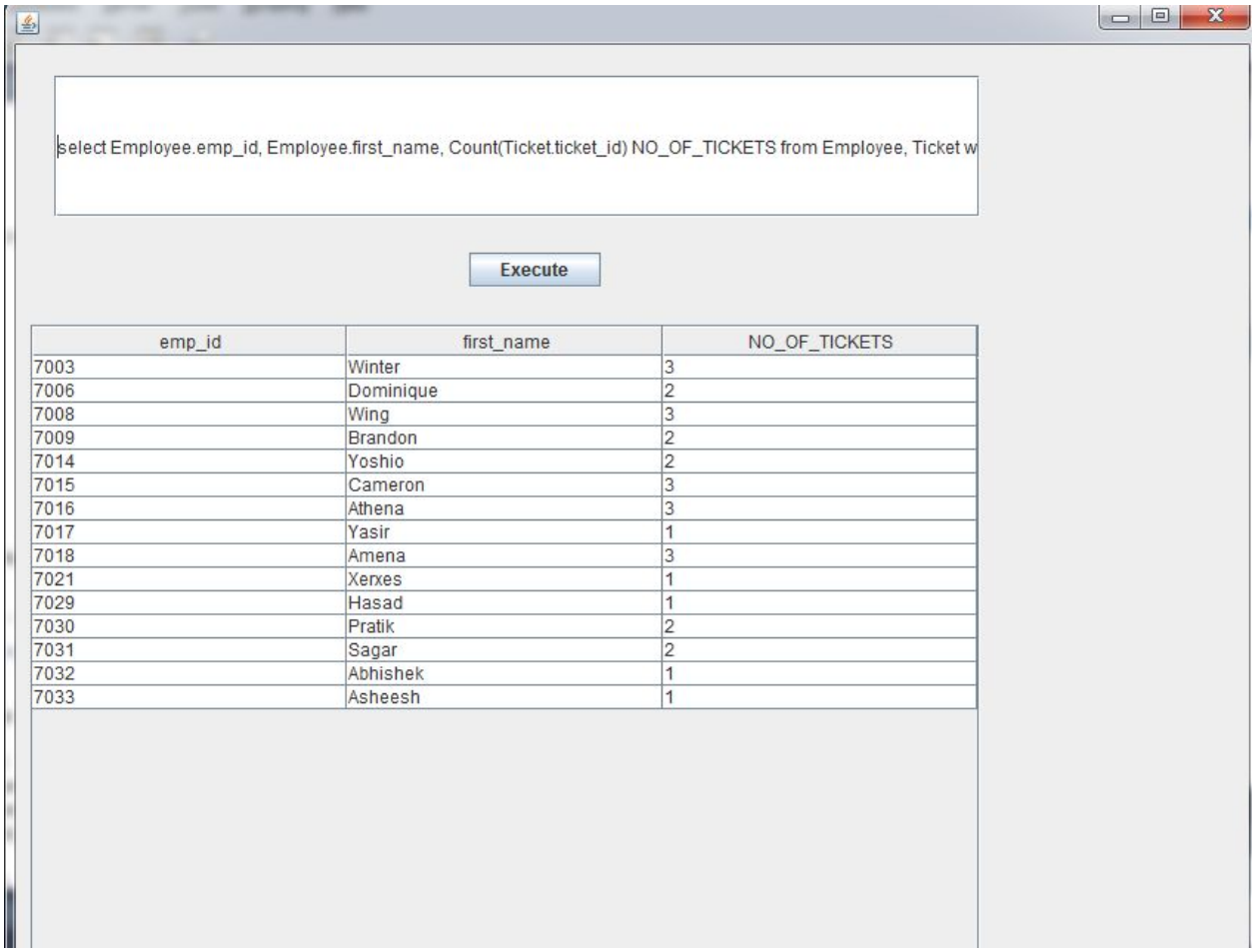
select Employee.emp_id, Employee.first_name from Employee,Department where Employee.manager_id = Department.manager_id and Department.department_name = 'Finance';



emp_id	first_name
7015	Cameron
7026	Kirk
7030	Pratik
7031	Sagar
7033	Asheesh

4)Display the count of tickets that were assigned to each employee.

select Employee.emp_id, Employee.first_name, Count(Ticket.ticket_id) NO_OF_TICKETS from Employee, Ticket where Ticket.assigned_to = Employee.emp_id group by Employee.emp_id;



emp_id	first_name	NO_OF_TICKETS
7003	Winter	3
7006	Dominique	2
7008	Wing	3
7009	Brandon	2
7014	Yoshio	2
7015	Cameron	3
7016	Athena	3
7017	Yasir	1
7018	Amena	3
7021	Xerxes	1
7029	Hasad	1
7030	Pratik	2
7031	Sagar	2
7032	Abhishek	1
7033	Asheesh	1

Sprint2

REQUIREMENTS

The goal is to create a ticketing tool system that caters the need of the business units i.e clients and IT service provider who is providing the service. The needs may include creating a ticket, assigning a ticket, checking the status of tickets, checking the SLA breach report, etc.
The updated user stories are as below:

User Stories	**Description**
US1	~~As an employee, I want to create/edit a ticket~~ As an employee of the IT contractor/Business Unit, I want to create a ticket in order to raise an issue.
US2	An employee logins/logout of account through which he/she will raise a ticket
US3	~~As a manager, I want to see the tickets assigned to a particular employee of my organisation~~ As a manager, I want to see the tickets assigned to my reportee of my department.
US4	As a manager, I want to see the tickets raised by particular employee of my organisation
US5	~~As a manager, I want to search the employee/employees whose skillset help solve a particular issue~~ As an Employee, I want to see the tickets assigned to me

US6	~~As an employee, I want to assign the ticket to another employee~~ As a employee of IT contractor/Business Unit, I want to assign the ticket to an employee after searching for relevant department concerned with the issue
US7	As a manager, I want to see the details of employee who have missed an SLA, so that I can access their performance
US8	As a manager, I want to search tickets based on date, SLA met/missed,topic
US9	As a manager, I want to see whether there are adequate human resource for a particular skill so that business needs can be maintained
US10	As a IT service provide, I want to store the information of services used by the various clients and relevant rates charged to them
US11	As a manager of the business unit/ IT service unit, I want to extract the billing information charged to various clients.

Note: The stories marked in orange have been completed in sprint 0. The stories marked in yellow have been taken care of in the current sprint.

CONCEPTUAL DESIGN

Entity:
[no change]
1)Employee

Attributes:
emp_id
name[composite]
first_name
last_name
phone_number[multi-valued]
email
designation

Primary Key justification: emp_id is taken as primary key since it is unique for every employee in the organisation. It is assumed that email id of an employee can change over time or with the transfer of employee to different location.

[no change]
2)Ticket

Attributes:
ticket_id
ticket_type
end_user_email
end_user_phone_number
severity
priority
opened_date
closed_date
status
description
resolution
age[Derived]

**end_user_email and end_user_phone_number are related to the customer of the business user (if any) who is facing an issue and have raised an issue to the business unit

Primary Key Justification: ticket_id is an autoincrement primary key generated and assigned to a ticket at the time of its creation.

[no change]
3)Account

Attributes:
email_id
password

Primary Key Justification: For every account of the employee, the employee will login with its email id

[no change]
3)Department

Attributes:
dept_id
department_name
department_code

Primary Key Justification: dept_id is the autoincrement primary key used to keep track of the department.
**department have been built based on the domain of the business unit egs for business units of oil and gas domain, there would be a separate department for oil and natural gas business units in the IT company

[modified]
4)BusinessUnit

Attributes:
business_unit_id
business_unit_name

Primary Key Justification: An auto increment primary key has been created to keep a track of the business units an IT company is working with

[no change]
5)Assignment Group

Attributes:

assignment_group_id

assignment_group_name

Primary Key Justification: An auto increment primary key has been created to keep a track of groups in the IT organisation.

[new entity]
6)Skill

Attributes:

skill_id

skill_name

Primary Key Justification: An auto increment primary key has been generated to keep track of the various skills relevant to the IT contractor

Relationships:
[no change]
1) Employee login/logout of Account

Cardinality: one to one

Participation: Employee has total participation; Account has total participation

[no change]
2) Employee creates Ticket

Cardinality : one to many

Participation: Employee has partial participation; Ticket has total participation

[no change]
3)Employee has Ticket

Cardinality: one to many

Participation: Employee has partial participation; Ticket has total participation

[no change]
4)Employee reports to Manager[Employee]

**this is a case of recursive relationship as manager is also an employee

Cardinality : many to one

Participation: Employee has total participation; Manager has partial participation

[no change]
5)Department has Assignment Group

Cardinality: one to many

Participation:Department has total participation; Assignment group has total participation

[no change]

6)Assignment Group has Employee

Cardinality: one to many

Participation: Assignment group has total participation; Employee has total participation

[no change]

7)Business Unit has Employee

Cardinality:One to Many

Participation: Business Unit has total participation; Employee has total participation

[no change]

8)Manager[Employee] manages the Department

Cardinality: one to one

Participation: Manager[Employee] has partial participation; Department has total participation

[no change]

9)Team Lead[Employee] manages Assignment Group

Cardinality: one to one

Participation: Team Lead[Employee] has partial participation; Assignment Group has a total participation.

[new relationship]

10)Employee has Skill

Cardinality: many to many

Participation: Employee has total participation; Skill has partial participation

[new relationship]

11) Ticket has an SLA

Cardinality: many to one

Participation: Ticket has total participation; SLA has total participation

LOGICAL DESIGN

Table: Employee

Columns:

<u>emp_id</u>

first_name

last_name

manager_id[foreign key;references emp_id of the employee table]

phone_personal

phone_business

email_id

designation

password

business_unit_id[foreign key; references business_unit_id of the business unit table]

Justification: email_id, password,business_unit_id could have been the separate

columns of the account table. But since it is total participation, we can include the columns in the employee table.

Normal Form: 2NF because a non-key attribute
email_id→ password

Breaking the table into two tables
[modified]
Table: Employee
emp_id
first_name
last_name
manager_id[foreign key;references emp_id of the employee table]
phone_personal
phone_business
designation
business_unit_id[foreign key; references business_unit_id of the business unit table]
Normal form: 4NF

[new table]
Table: Account
emp_id
email_id
password

Normal Form: 4NF

[no change]
Table: Ticket
Columns:
ticket_id
ticket_type
end_user_email
end_user_phone_number
severity
priority
opened_date
closed_date
opened_by[foreign key; references emp_id of the employee table]
status
description
resolution

assigned_to[foreign key; references emp_id of the employee table]

Normal Form:4NF

[no change]
Table: Department
Columns:

department_id
department_name
department_code
manager_id[foreign key;references emp_id of the employee table]

Normal Form: 2NF; because the non-key attribute manager_id → department_name,department_code and department_code → manager_id

The table has not been decomposed into 4NF because that will require making a separate table department_manager with department_id and manager_id as primary key. This will create too many joins and make the design complex as one extra join will be required to fetch the data of the manager.

[no change]
Table: BusinessUnit
Columns:

business_unit_id
business_unit_name

Normal form: 4NF

[no change]
Table: AssignmentGroup
Columns:

group_id
group_name
department_id[foreign key;references department_id from department]
team_lead_id[foreign key;references emp_id from employee]

Normal Form: 2NF; The table is in 2NF team_lead_id → group_name. However this table has

not been broken down into 3NF because in the current application context this will create complexity and one extra join will be required to fetch the data of the teamlead.

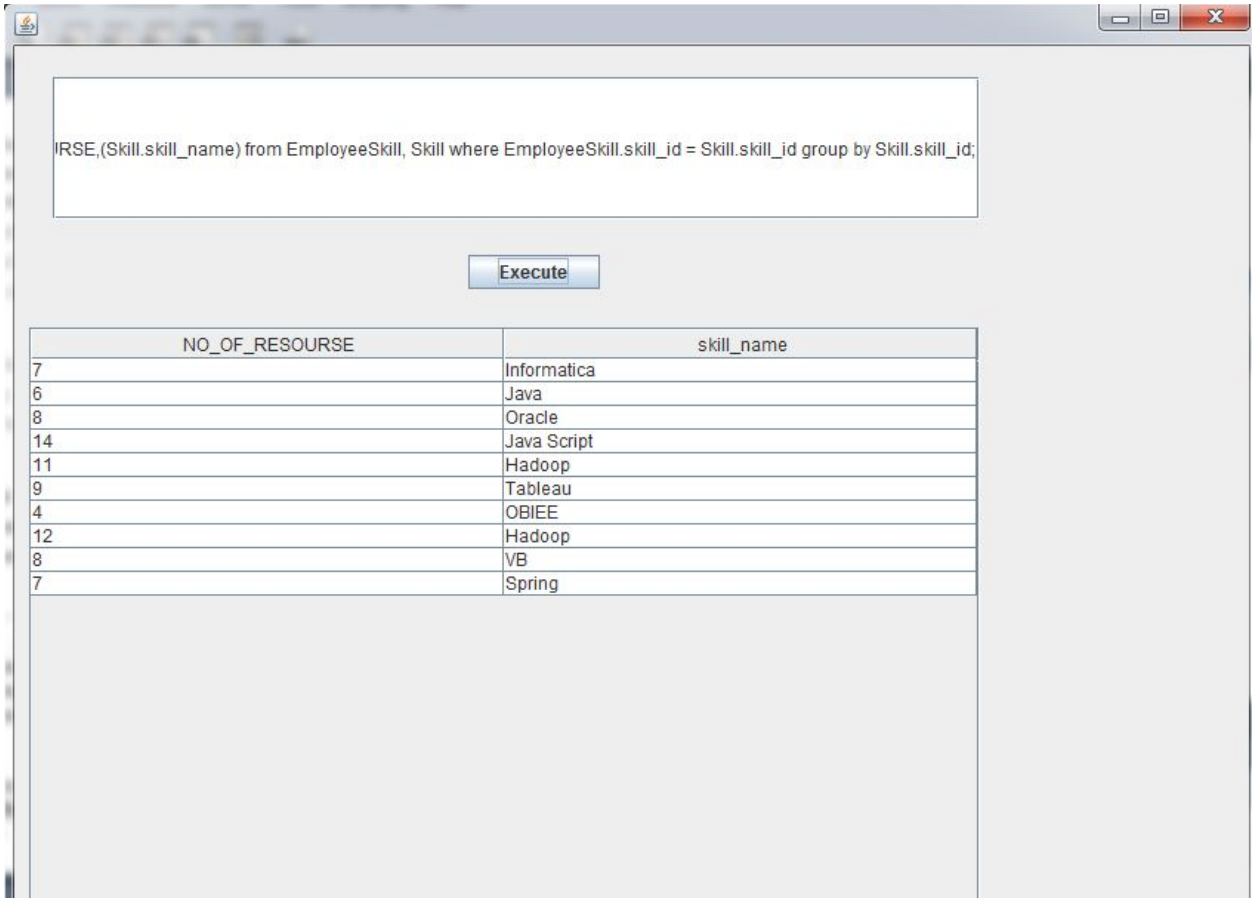
[new table]
Table: Skill
Columns:
skill_id
skill_name
Normal Form: 4NF
Primary Key Justification: skill_id is the primary key to identify each row uniquely in the table

[new table]
Table: SLAInfo
Columns:
severity[Foreign Key;References severity of the Ticket Table]
ticket_type[Foreign Key;References ticket_type of the Ticket Table]
permissible_age
Normal Form:4NF
Primary Key Justification: severity and ticket_type uniquely determines a permissible age of a ticket.

Queries:

1)Query to get the count of resources of various skills in the Company

```
select count(EmployeeSkill.emp_id)  NO_OF_RESOURSE,(Skill.skill_name) from
EmployeeSkill, Skill where EmployeeSkill.skill_id = Skill.skill_id group by Skill.skill_id;
```

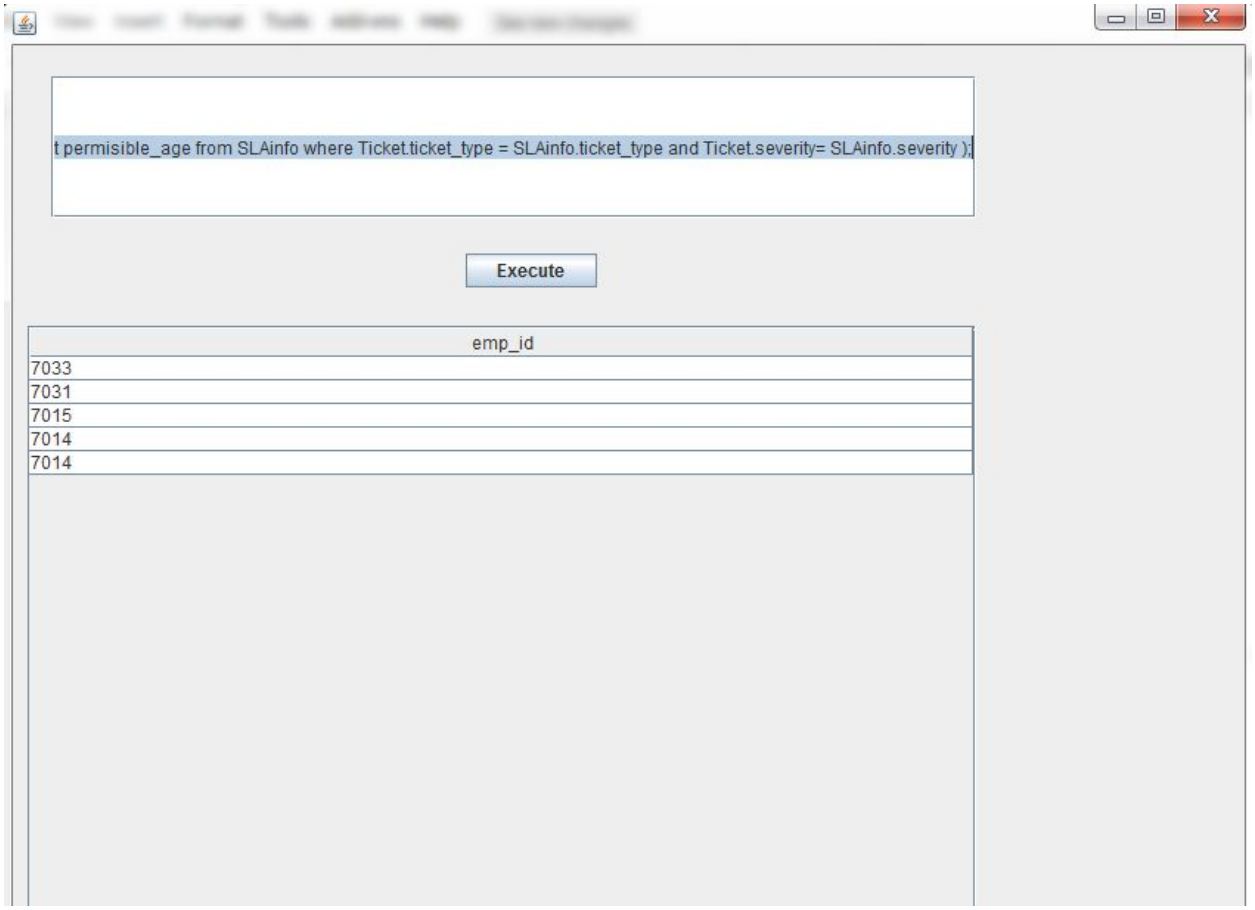


NO_OF_RESOURSE	skill_name
7	Informatica
6	Java
8	Oracle
14	Java Script
11	Hadoop
9	Tableau
4	OBIEE
12	Hadoop
8	VB
7	Spring

2)Query to get the employee_id who have missed the SLA

select Employee.emp_id from Employee,Ticket where Ticket.status = 'Resolved' and Ticket.assigned_to = Employee.emp_id and (DATE(Ticket.closed_date) - DATE(Ticket.opened_date))> (select permisible_age from SLAinfo where Ticket.ticket_type =

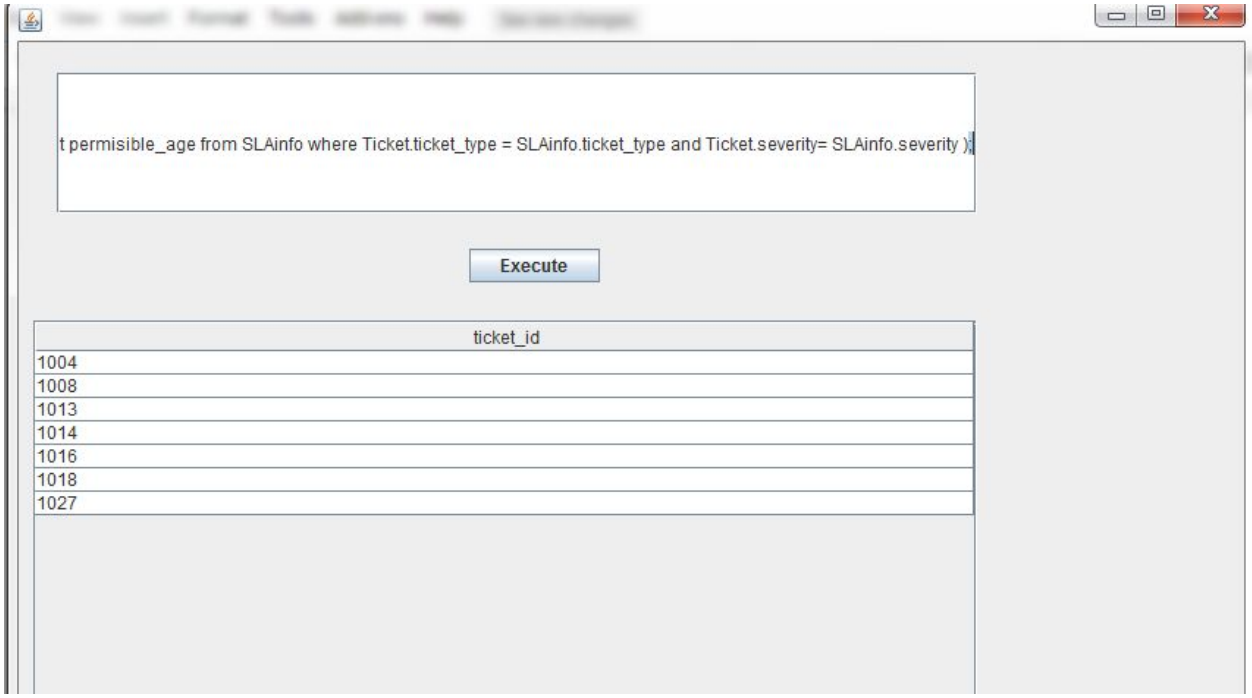
SLAinfo.ticket_type and Ticket.severity= SLAinfo.severity);



3)Query to get the Tickets that have not been closed even though there SLA has been missed

select Ticket.ticket_id from Ticket where Ticket.status = 'open' and (DATE(NOW()) - DATE(Ticket.opened_date))>
(select permisible_age from SLAinfo where Ticket.ticket_type = SLAinfo.ticket_type and

Ticket.severity= SLAinfo.severity);



Sprint3

REQUIREMENTS

The goal is to create a ticketing tool system that caters the need of the business units i.e clients and IT service provider who is providing the service. The needs may include creating a ticket, assigning a ticket, checking the status of tickets, checking the SLA breach report, etc.
The updated user stories are as below:

User Stories	Description
US1	As an employee of the IT contractor/Business Unit, I want to create a ticket in order to raise an issue.
US2	An employee logins/logout of account through which he/she will raise a ticket
US3	As a manager, I want to see the tickets assigned to my reportee of my department.
US4	As a manager, I want to see the tickets raised by particular employee of my organisation
US5	As an Employee, I want to see the tickets assigned to me
US6	As a employee of IT contractor/Business Unit, I want to assign the ticket to an employee after searching for relevant department concerned with the issue
US7	As a manager, I want to see the details of employee who have missed an SLA, so that I can access their performance

US8	As a manager, I want to search tickets based on date, SLA met/missed,topic
US9	As a manager, I want to see whether there are adequate human resource for a particular skill so that business needs can be maintained
US10	As a IT service provide, I want to store the information of services used by the various clients and relevant rates charged to them
US11	As a manager of the business unit/ IT service unit, I want to extract the billing information charged to various clients.

Note: The stories marked in orange have been completed in sprint 1. The stories marked in yellow have been taken care of in the sprint2. The stories marked in cyan have been taken care of in the current sprint.

CONCEPTUAL DESIGN

Entity:
[no change]
1)Employee
- Attributes:
- <u>emp_id</u>
- name[composite]
  - first_name
  - last_name
- phone_number[multi-valued]
- email
- designation

Primary Key justification: emp_id is taken as primary key since it is unique for every employee in the organisation. It is assumed that email id of an employee can change over time or with the transfer of employee to different location.

[modified]
2)Ticket

Attributes:
<u>ticket_id</u>
ticket_type
end_user_email
end_user_phone_number
severity
~~priority~~
opened_date
closed_date
status
description
resolution
last_modified_date_time[new column]
age[Derived]

**end_user_email and end_user_phone_number are related to the customer of the business user (if any) who is facing an issue and have raised an issue to the business unit

Primary Key Justification: ticket_id is an autoincrement primary key generated and assigned to a ticket at the time of its creation.

[no change]
3)Account

Attributes:
<u>email_id</u>
password

Primary Key Justification: For every account of the employee, the employee will login with its email id(Refer the logical design for better primary key alternative). Here in the database, emp_id is kept as primary key because though possibility is rare, the email_id can change. One scenario is company decides to change it’s domain name.

[no change]
3)Department

Attributes:
department_name
<u>department_code</u>

Primary Key Justification: department_code is the primary key used to keep track of the department.
**department have been built based on the domain of the business unit egs for business units of oil and gas domain, there would be a separate department for oil and natural gas business units in the IT company. Though department_code is a good candidate for primary key there is a possibility that department code changes for each department depending upon the domain the IT company is targeting. So an auto-increment department_id is better suited.

[modified]
4)BusinessUnit

Attributes:
business_unit_name
business_base_rate[new attribute]

Primary Key Justification: An auto increment primary key has been created to keep a track of the business units an IT company is working with. The business_unit_name does forms a unique entity but code also change over time. For eg. a comany changing its name for rebranding.

[no change]
5)Assignment Group

Attributes:
assignment_group_name

Primary Key Justification: An auto increment primary key has been created to keep a track of groups in the IT organisation.

[no change]
6)Skill

Attributes:
skill_name

Primary Key Justification: An auto increment primary key has been generated to keep track of the various skills relevant to the IT contractor

[modified]
7)SLAInfo

Attributes:
<u>severity</u>
<u>ticket_type</u>
permissible_age
ticket_base_rate[new attribute]

Primary Key Justification: The severity and ticket_type are the attributes that constitute unique set of permissible age and ticket rate values

[new entity]
8)TicketActivityLog

Attributes:
<u>ticket_id</u>
activity
past_value
new_value
<u>time_stamp</u>

Primary Key Justification:At any point of time, no two changes can occur on a single ticket. Thus, ticket_id and time_stamp forms the primary key.

Relationships:
[no change]
1) Employee login/logout of Account

Cardinality: one to one
Participation: Employee has total participation; Account has total participation

[no change]
2) Employee creates Ticket

Cardinality : one to many
Participation: Employee has partial participation; Ticket has total participation

[no change]
3)Employee has Ticket

Cardinality: one to many
Participation: Employee has partial participation; Ticket has total participation

[no change]
4)Employee reports to Manager[Employee]

**this is a case of recursive relationship as manager is also an employee
Cardinality : many to one
Participation: Employee has total participation; Manager has partial participation

[no change]

5)Department has Assignment Group

Cardinality: one to many

Participation:Department has total participation; Assignment group has total participation

[no change]

6)Assignment Group has Employee

Cardinality: one to many

Participation: Assignment group has total participation; Employee has total participation

[no change]

7)Business Unit has Employee

Cardinality:One to Many

Participation: Business Unit has total participation; Employee has total participation

[no change]

8)Manager[Employee] manages the Department

Cardinality: one to one

Participation: Manager[Employee] has partial participation; Department has total participation

[no change]

9)Team Lead[Employee] manages Assignment Group

Cardinality: one to one

Participation: Team Lead[Employee] has partial participation; Assignment Group has a total participation.

[no change]

10)Employee has Skill

Cardinality: many to many

Participation: Employee has total participation; Skill has partial participation

[no change]

11) Ticket has an SLA

Cardinality: many to one

Participation: Ticket has total participation; SLA has total participation

LOGICAL DESIGN WITH HIGHEST NORMAL FORMS AND INDEXES

[no change]

1)Table: Employee

emp_id
first_name
last_name
manager_id[foreign key;references emp_id of the employee table]
phone_personal
phone_business
designation
business_unit_id[foreign key; references business_unit_id of the business unit table]

Normal form: 4NF

Indexes :

Columns	Type	Justification
emp_id	clustered	As it is the primary key, it is auto created
reports_to	non clustered	As it is the Foreign key, it is auto created
business_unit_id	non clustered	As it is the Foreign key, it is auto created

[new table]
Table: Account
emp_id
email_id
password
Normal Form: 4NF
Indexes :

Column	Type	Justification
email_id	non-clustered	yes;Since it is unique constraint was specified at table creation index was created
emp_id(Primary key)	clustered	As emp_id is the primary key, index is automatically created

[no change]
Table: Ticket
Columns:

- ticket_id
- ticket_type
- end_user_email
- end_user_phone_number
- severity
- priority
- opened_date
- closed_date
- opened_by[foreign key; references emp_id of the employee table]
- status
- description
- resolution
- last_modified_date_time[new column]
- assigned_to[foreign key; references emp_id of the employee table]

Normal Form:4NF
Indexes :

Column	Type	Justification
ticket_id	Clustered	As it is the primary key, it is auto created
ticket_type,severity	non-clustered	As it is the Foriegn key, it is auto created
opened_by	non-clustered	As it is the Foriegn key, it is auto created
opened_date	non-clustered	Index is created a ticket is often queried using the opened date parameter
closed_date	non-clustered	Index is created a ticket is often queried using the closed date parameter

status	non-clustered	Status of tickets are often queried. As it is e num, full text index can't be created
assigned_to	non-clustered	As it is the Foreign key, it is auto created

[no change]
Table: Department
Columns:

department_id
department_name
department_code
manager_id[foreign key;references emp_id of the employee table]

Normal Form: 2NF; because the non-key attribute manager_id → department_name,department_code and department_code → manager_id
The table has not been decomposed into 4NF because that will require making a separate table department_manager with department_id and manager_id as primary key. This will create too many joins and make the design complex as one extra join will be required to fetch the data of the manager.

Indices:

Column	Type	Justification
department_id	clustered	As it is the primary key, it is auto created
manager_id	non-clustered	As it is the Foreign key, it is auto created
department_code	non-clustered	User query often by department code as they remember 4 letter code easily

[modified]
Table: BusinessUnit
Columns:

business_unit_id
business_unit_name
business_base_rate

Normal form: 4NF
Indices:

Column	Type	Justification
business_unit_id	clustered	As it is the primary key, it is auto created
business_unit_name	Non clustered	Employees can query something by using business unit name

[no change]
Table: AssignmentGroup
Columns:

group_id
group_name
department_id[foreign key;references department_id from department]
team_lead_id[foreign key;references emp_id from employee]

Normal Form: 2NF; The table is in 2NF team_lead_id → group_name. However this table has not been broken down into 3NF because in the current application context this will create complexity and one extra join will be required to fetch the data of the team lead.

Indices:

Column	Type	Justification
group_id	clustered	As it is the primary key, it is auto created
department_id	non-clustered	As it is the Foriegn key, it is auto created
team_lead_id	non-clustered	As it is the Foreign key, it is auto created

[no change]
Table: Skill
Columns:
skill_id
skill_name
Normal Form: 4NF
Primary Key Justification: skill_id is the primary key to identify each row uniquely in the table

Indices:

Columns	Type	Justification
skill_id	clustered	As it is the primary key, it is auto created

[modified]
Table: SLAInfo
Columns:
severity[Foreign Key;References severity of the Ticket Table]
ticket_type[Foreign Key;References ticket_type of the Ticket Table]
permissible_age
ticket_base_rate[new column]
Normal Form:4NF
Primary Key Justification: severity and ticket_type uniquely determines a permissible age of a ticket.

Indices:

Columns	Type	Justification
Severity,ticket_type	Clustered	As it is the primary key, it is auto created

[new table]
Table: TicketActivityLog
Columns:

ticket_id
activity
past_value
new_value
time_stamp

Primary Key Justification: A ticket can experience only one change at any point of time. Thus combination of both uniquely determines every row.
Normal Form:4NF
Indices:

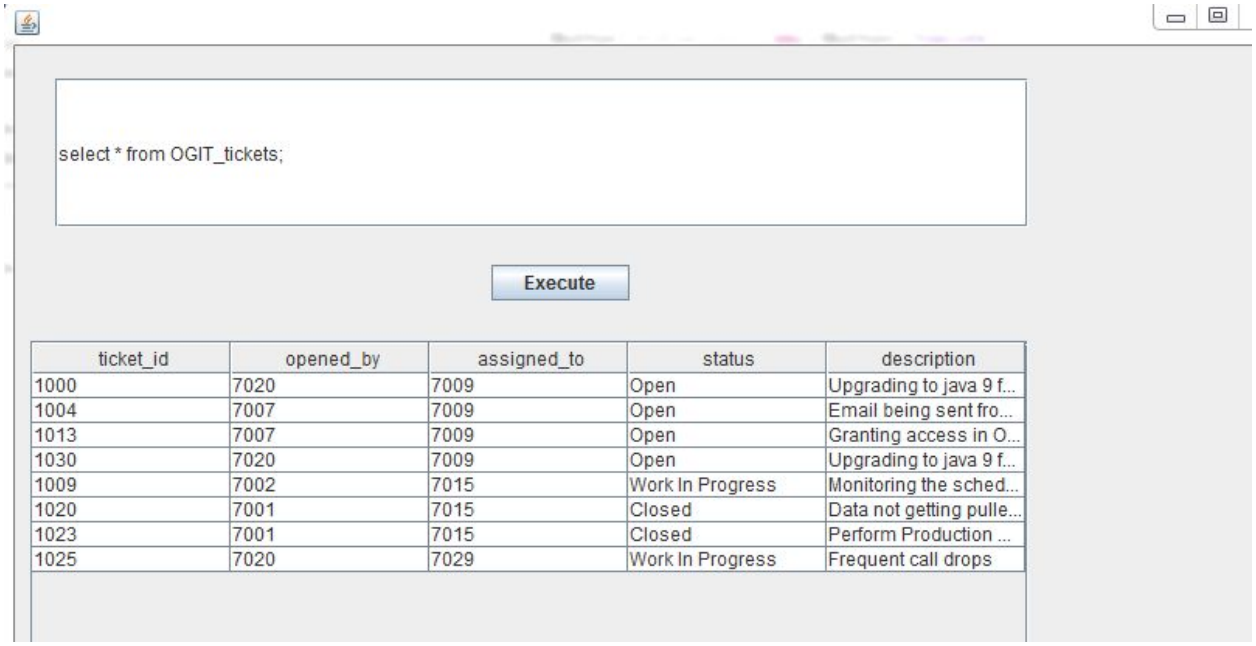
Column	Type	Justification
ticket_id,time_stamp	clustered	As it is the primary key, it is auto created

VIEWS AND STORED PROGRAMS

Views: *Manager can view all the tickets assigned to his/her department*
Associated View:*OGIT_tickets, FSDC_tickets, SCIT_tickets, TLIT_tickets*

Goal: The main purpose of this view is to keep manager level view where he/she would be able to view all tickets assigned to his/her department so as to keep a track of pending tickets.

Sample ScreenShots:

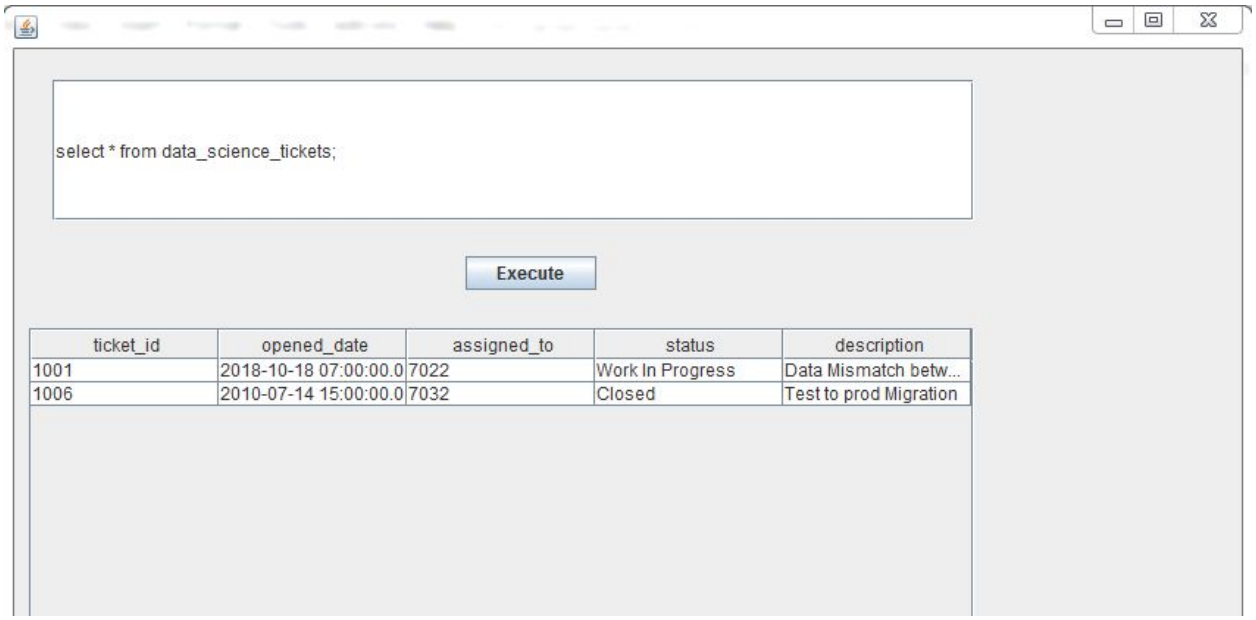


ticket_id	opened_by	assigned_to	status	description
1000	7020	7009	Open	Upgrading to java 9 f...
1004	7007	7009	Open	Email being sent fro...
1013	7007	7009	Open	Granting access in O...
1030	7020	7009	Open	Upgrading to java 9 f...
1009	7002	7015	Work In Progress	Monitoring the sched...
1020	7001	7015	Closed	Data not getting pulle...
1023	7001	7015	Closed	Perform Production ...
1025	7020	7029	Work In Progress	Frequent call drops

View: *Team Lead can view all the tickets assigned to his/her assignment group*
Associated View:*app_development_tickets, intelligent_apps_tickets, visualisation_tickets, big_data_infrastructure, ios_app_development_tickets, bi_tools_tickets, data_science_tickets, infrastructure_tickets, cloud_tickets, tel_infra_tickets.*

Goal: The main purpose of this view is to keep Team Lead level view where he/she can view the tickets assigned to his/her team or to team members so as to keep a track for individual employee work.

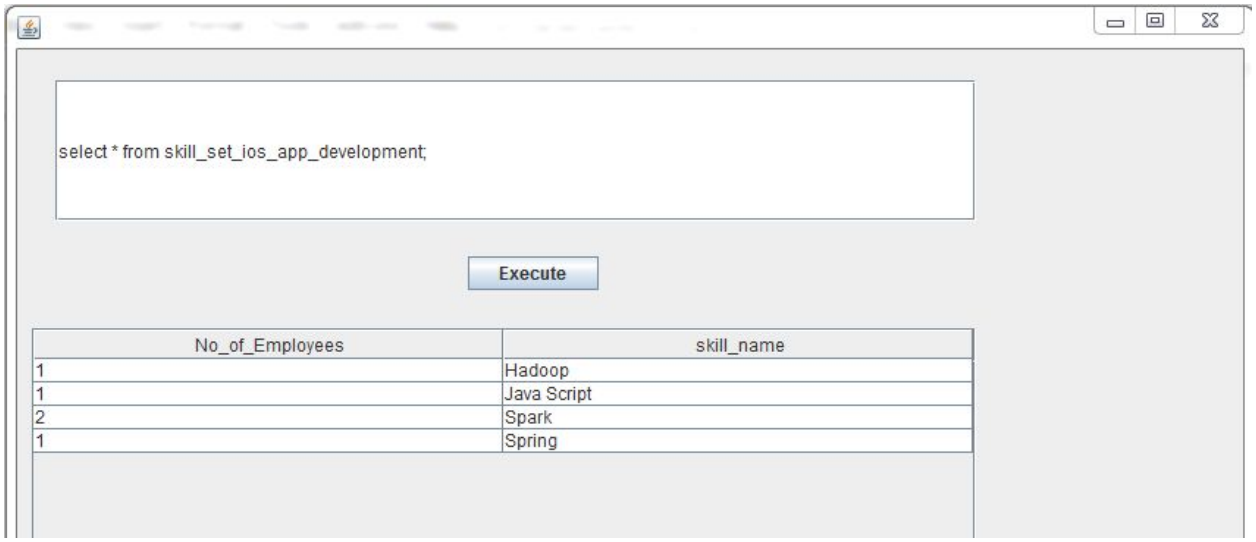
Sample Screenshot:



View: *Team Lead can see the skill set distribution of Assignment Group*
Associated View: *skill_set_dist_app_development, skill_set_dist_bi_tools,skill_set_dist_infrastructure, skill_set_dist_visualization, skill_set_dist_intelligent_apps, skill_set_dist_cloud, skill_set_dist_big_data_infrastructure, skill_set_dist_data_science, skill_set_ios_app_development, skill_set_telecom_infrastructure*

Goal: The main purpose for this view is to again keep a different team lead level view where he/she can view the distribution of the employee in a particular assignment group depending on their skill sets and if there are lack of resources in the team for a particular skill and the necessary actions could be taken

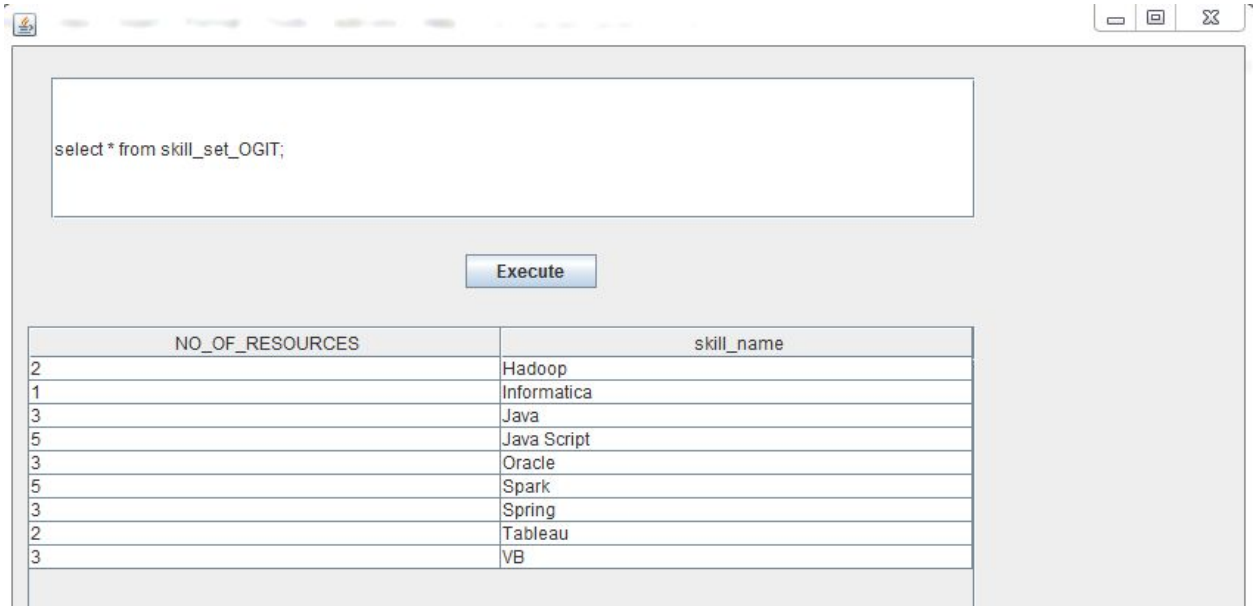
Sample Screenshot:



No_of_Employees	skill_name
1	Hadoop
1	Java Script
2	Spark
1	Spring

View: *Manager can see the skill set distribution of his/her department*
Associated View: *skill_set_OGIT, skill_set_TLIT, skill_set_FSDC, skill_set_SCIT*

Goal: The main purpose for this view is to again keep a different manager level view where he/she can view the distribution of the employee in his department depending upon the skill sets and can decide if any actions are required for the same.
Sample Screenshot:



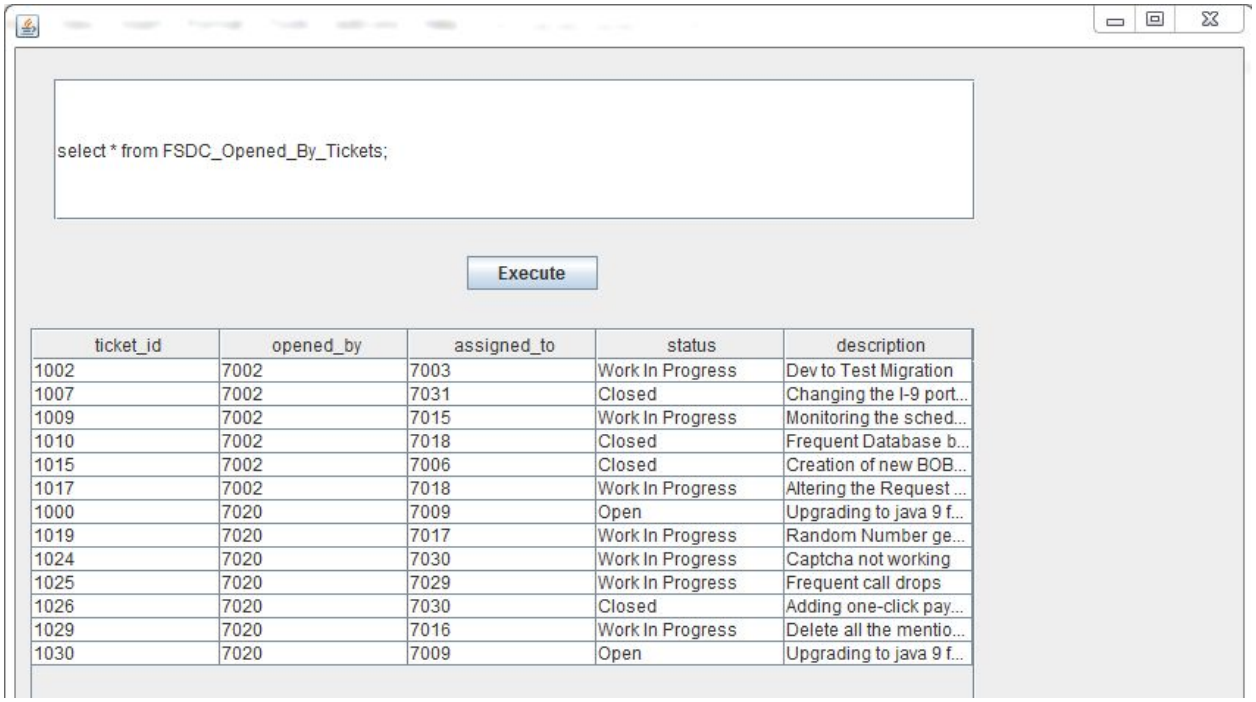
NO_OF_RESOURCES	skill_name
2	Hadoop
1	Informatica
3	Java
5	Java Script
3	Oracle
5	Spark
3	Spring
2	Tableau
3	VB

View: *Manager see the ticket raised by employee of his/her department*

Associated View: *OGIT_Opened_By_Tickets, SCIT_Opened_By_Tickets, TLIT_Opened_By_Tickets, FSDC_Opened_By_Tickets*

Goal: The main purpose for this view is to keep a different manager level view which depends on if employee from his/her department have raised a ticket or not. If any one from respective department has raised the ticket, view can help a manger to keep a track of that ticket if it requires any escalation from manager at certain point of time.

Sample Screenshot:



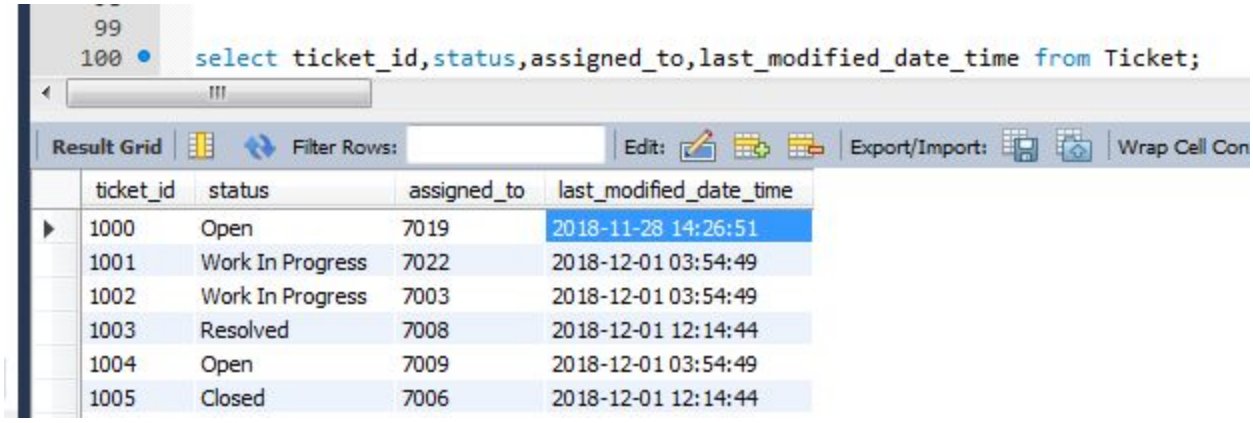
ticket_id	opened_by	assigned_to	status	description
1002	7002	7003	Work In Progress	Dev to Test Migration
1007	7002	7031	Closed	Changing the I-9 port...
1009	7002	7015	Work In Progress	Monitoring the sched...
1010	7002	7018	Closed	Frequent Database b...
1015	7002	7006	Closed	Creation of new BOB...
1017	7002	7018	Work In Progress	Altering the Request ...
1000	7020	7009	Open	Upgrading to java 9 f...
1019	7020	7017	Work In Progress	Random Number ge...
1024	7020	7030	Work In Progress	Captcha not working
1025	7020	7029	Work In Progress	Frequent call drops
1026	7020	7030	Closed	Adding one-click pay...
1029	7020	7016	Work In Progress	Delete all the mentio...
1030	7020	7009	Open	Upgrading to java 9 f...

Stored procedure:

1)*autoAssignmentTicket*:

Parameters: None

Goal: The goal of this procedure is to auto assign the unattended tickets to an employee's superior. If the ticket is in 'Open' state for more than 48 hours and no action has been taken on it it will automatically assigned to the an employee's superior.



ticket_id	status	assigned_to	last_modified_date_time
1000	Open	7019	2018-11-28 14:26:51
1001	Work In Progress	7022	2018-12-01 03:54:49
1002	Work In Progress	7003	2018-12-01 03:54:49
1003	Resolved	7008	2018-12-01 12:14:44
1004	Open	7009	2018-12-01 03:54:49
1005	Closed	7006	2018-12-01 12:14:44

Superior of 7019 is 7009.

```
select ticket_id,status,assigned_to,last_modified_date_time from Ticket;
```

emp_id	first_name	last_name	reports_to	phone_personal	phone_business	designation	business_unit_
7019	Vivien	Joseph	7009	15688669797	15677989999	Graduate Engineer	5000
7029	Hasad	Phillips	7009	12394634956	16435943999	Graduate Engineer	5000
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

After 1 min, the ticket is assigned to 7009

```
select ticket_id,status,assigned_to,last_modified_date_time from Ticket;
```

ticket_id	status	assigned_to	last_modified_date_time
1000	Open	7009	2018-12-02 00:04:51
1001	Work In Progress	7022	2018-12-01 03:54:49
1002	Work In Progress	7003	2018-12-01 03:54:49
1003	Resolved	7008	2018-12-01 12:14:44
1004	Open	7009	2018-12-01 03:54:49

2)*autoClosureTicket*

Parameters:None

Goal: the goal of this procedure is to auto close the ticket if the ticket is in resolved state and has been in the same state since past 48 hours.

ticket_id 1003 was last modified on 28-Nov-2018

```
select ticket_id,status,assigned_to,last_modified_date_time from Ticket;
```

ticket_id	status	assigned_to	last_modified_date_time
1000	Open	7009	2018-12-02 00:04:51
1001	Work In Progress	7022	2018-12-01 03:54:49
1002	Work In Progress	7003	2018-12-01 03:54:49
1003	Resolved	7008	2018-11-28 12:14:44

After 1 min, ticket 1003 goes to closed state

```
select ticket_id,status,assigned_to,last_modified_date_time from Ticket;
```

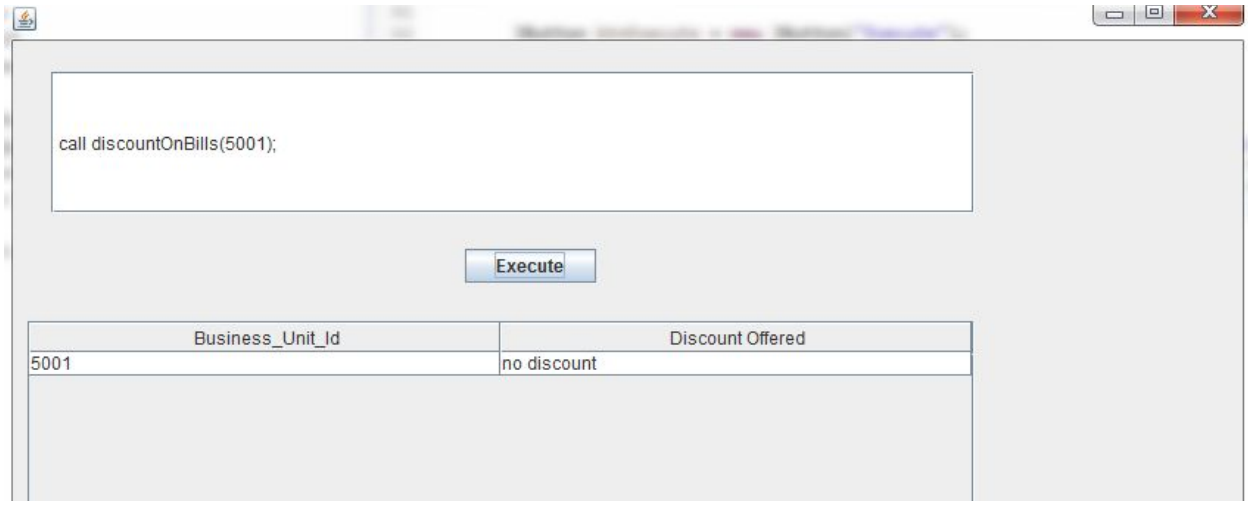
Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Conten

ticket_id	status	assigned_to	last_modified_date_time
1000	Open	7009	2018-12-02 00:04:51
1001	Work In Progress	7022	2018-12-01 03:54:49
1002	Work In Progress	7003	2018-12-01 03:54:49
1003	Closed	7008	2018-12-02 00:10:25
1004	Open	7009	2018-12-01 03:54:49

3)*discountOnBills*

Parameters:in business_unit_id int

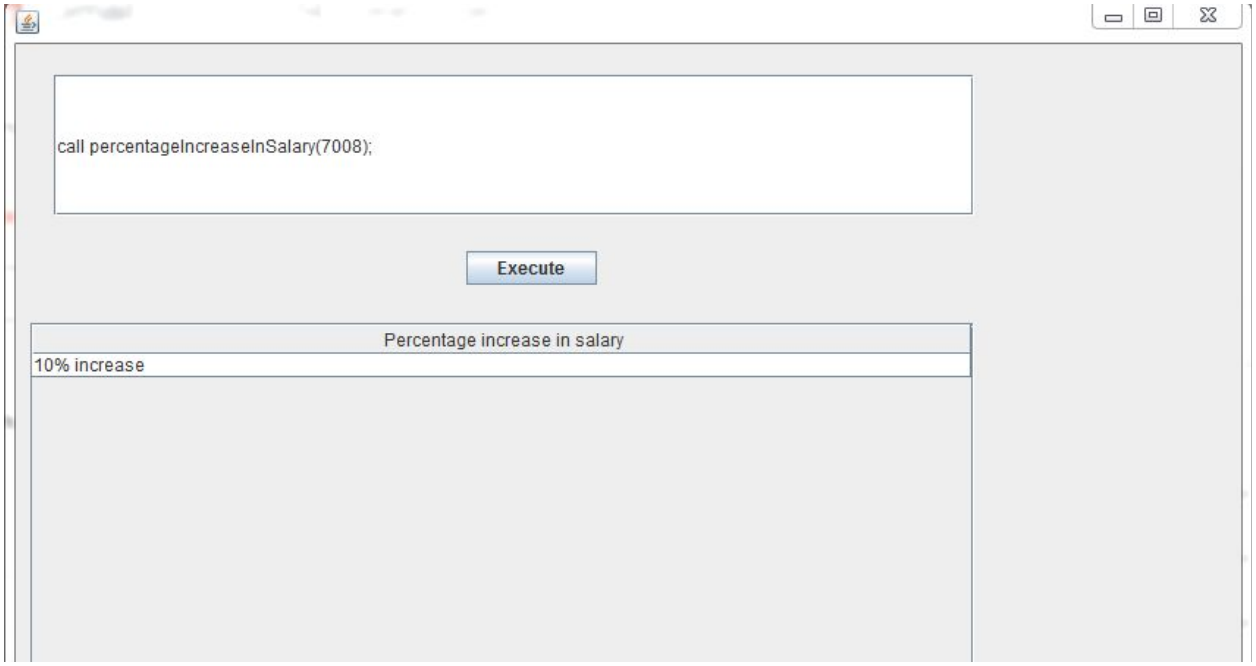
Goal:To offer discount on bill to particular business unit according to number of SLA missed



4)*percentageIncreaseInSalary*

Parameters:in emp_id int

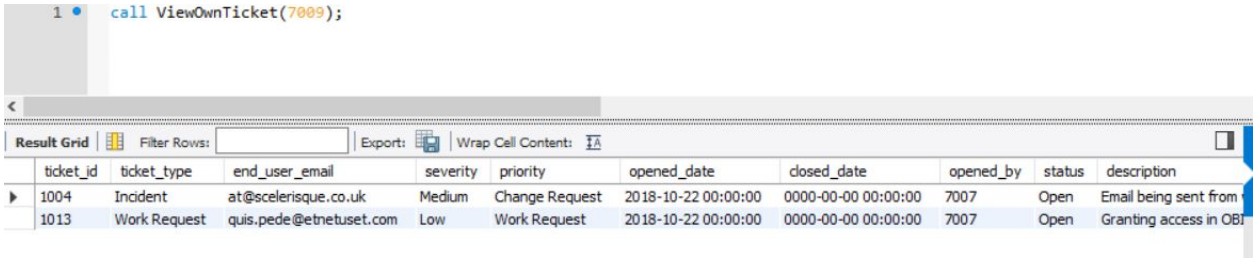
Goal:To calculate percentage increase in the salary of the employee based on his/her performance (depending on number of tickets resolved in a month)



5)*viewOwnTicket*

Parameters: in emp_id int

Goal:To permit an employee to view tickets assigned to him/her which are not yet resolved



```
1 • call ViewOwnTicket(7009);
```

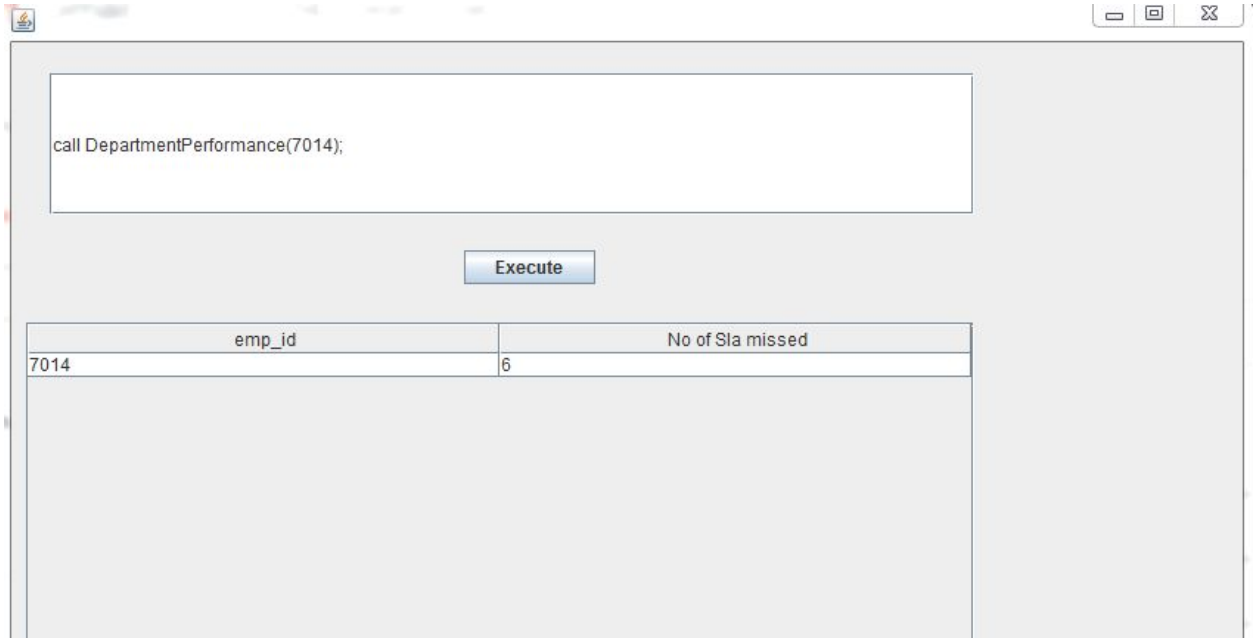
Result Grid | Filter Rows: | Export: | Wrap Cell Content:

ticket_id	ticket_type	end_user_email	severity	priority	opened_date	closed_date	opened_by	status	description
1004	Incident	at@scelerisque.co.uk	Medium	Change Request	2018-10-22 00:00:00	0000-00-00 00:00:00	7007	Open	Email being sent from
1013	Work Request	quis.pede@etnetuset.com	Low	Work Request	2018-10-22 00:00:00	0000-00-00 00:00:00	7007	Open	Granting access in OB

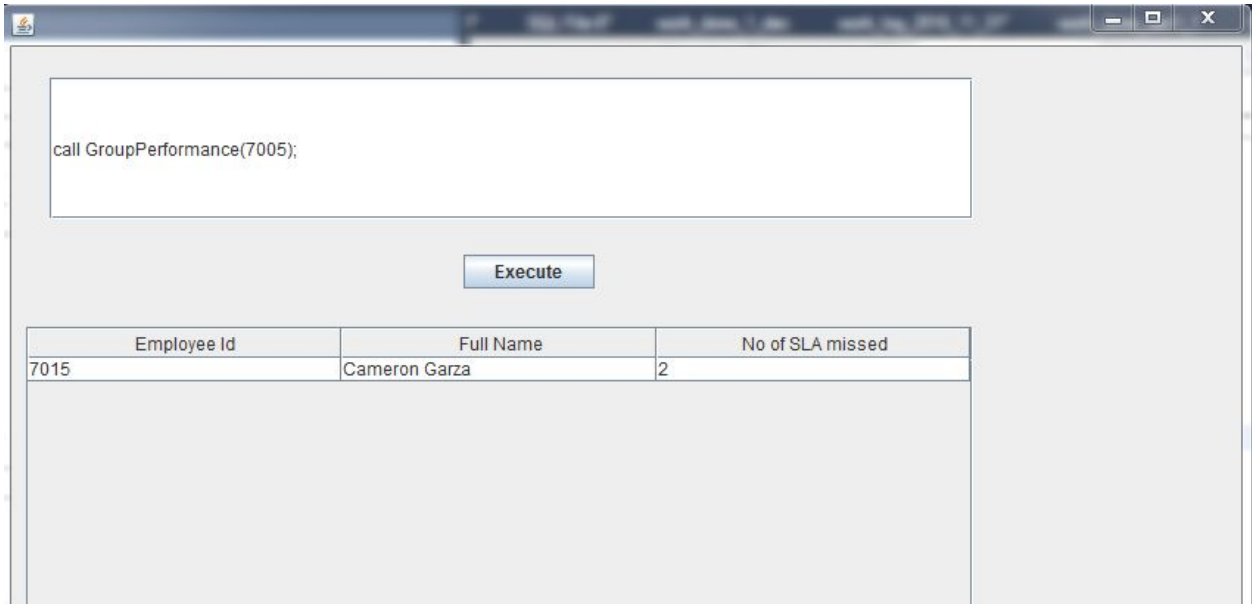
6)*departmentPerformance*

Parameters:in manager_id int

Goal: To permit a manager to see the performance of the employees belonging to his/her department (gives count of SLA missed by each employee)



7)*groupPerformance*
Parameters:in team_lead_id int
Goal:To permit a Team Lead to see the performance of the employees belonging to his/her assignment group (gives count of SLA missed by each employee)



8) *resolvedTicketRaisedByOwn*
Parameters: in emp_id int
Goal: To permit an employee to view the tickets resolved by him/her

```
call ResolvedTicketRaisedByOwn(7008);
```

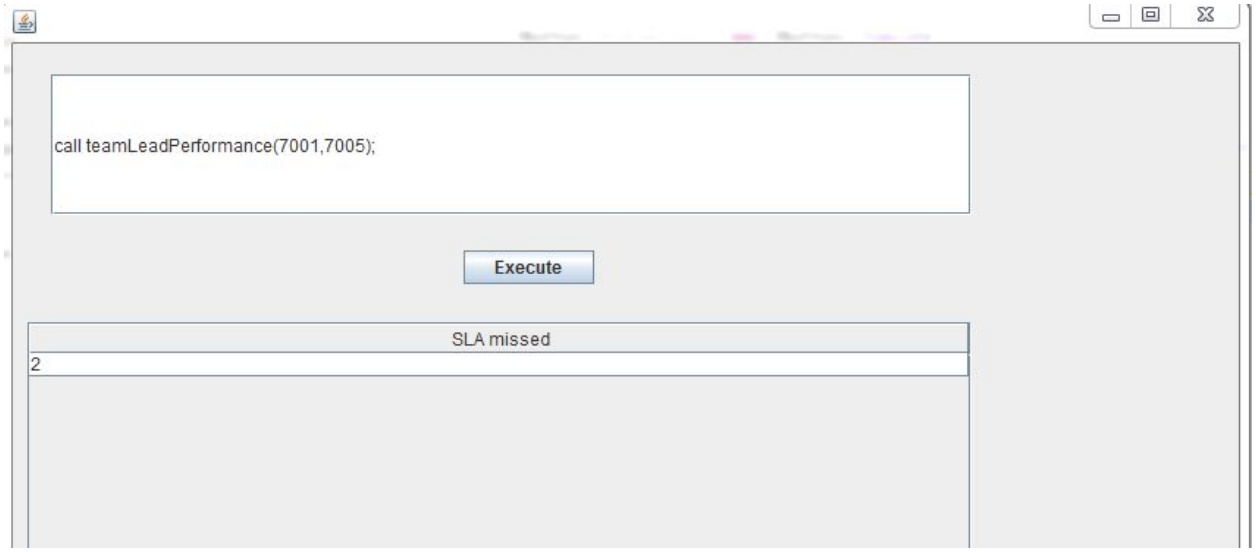
Result Grid | Filter Rows: | Export: | Wrap Cell Content:

ticket_id	ticket_type	end_user_email	severity	priority	opened_date	closed_date	opened_by	status
1000	Incident	elit@elit.co.uk	Low	Change Request	2010-09-04 00:00:00	2010-09-18 00:00:00	7020	Resolved
1019	Incident	Etiam.imperdiet.dictum@erat.edu	High	Work Request	2018-10-13 00:00:00	2017-09-10 00:00:00	7020	Work In Progress
1024	Incident	Nullam.feugiat.placerat@mollis.co.uk	Low	Work Request	2018-10-12 00:00:00	0000-00-00 00:00:00	7020	Work In Progress
1025	Problem	nisi.Aenean@Proinnon.com	Low	Work Request	2018-10-11 00:00:00	0000-00-00 00:00:00	7020	Work In Progress
1026	Change Request	pede.ultrices@erosNam.ca	High	Change Request	2017-09-15 00:00:00	2017-09-15 00:00:00	7020	Resolved
1029	Work Request	arcu.Morbi.sit@eratVivamusnisi.com	Medium	Work Request	2018-10-09 00:00:00	0000-00-00 00:00:00	7020	Work In Progress

9) *teamLeadPerformance*

Parameters: in manager_id int, in team_lead_id int

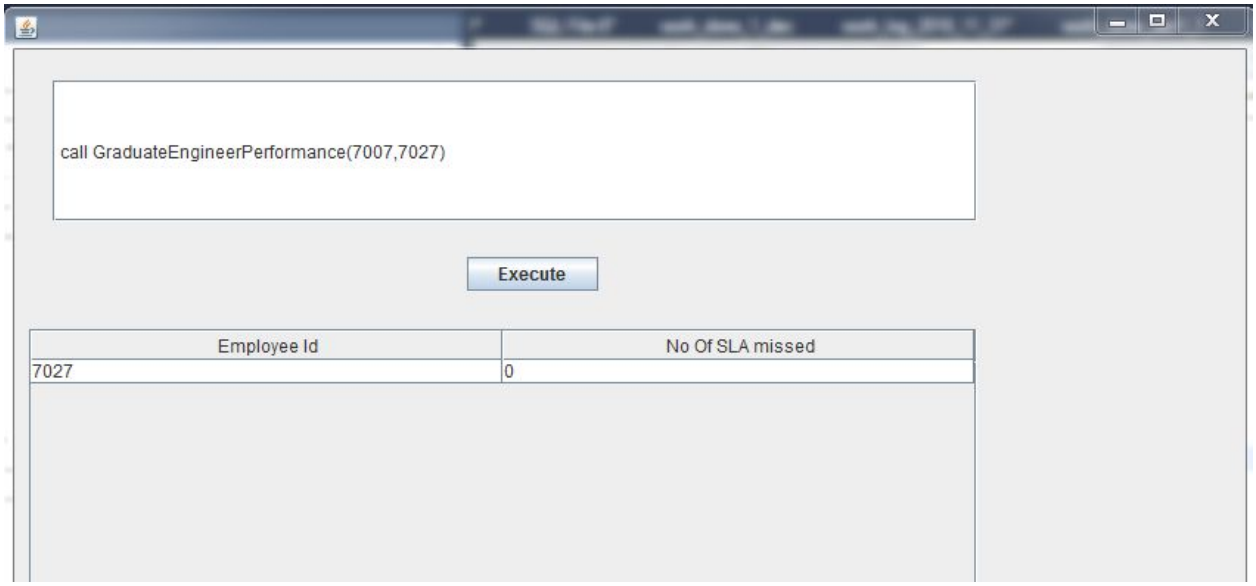
Goal: To permit a manager to see the performance for a particular Team lead reporting to him



10)*graduateEngineerPerformance*

Parameters:in manager_id int,in emp_id int

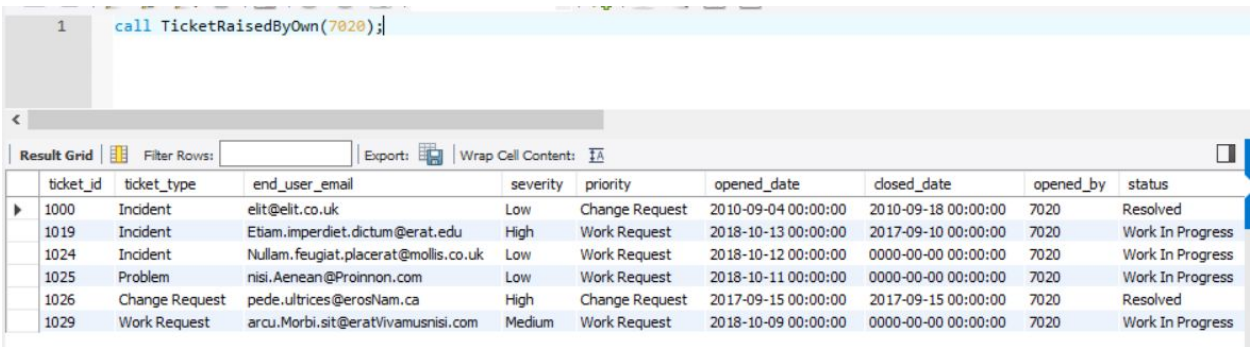
Goal: To permit a Team Lead to see the performance of the Graduate engineer reporting to him/her (gives count of SLA missed by each Engineer)



11) *TicketRaisedByOwn*

Parameters: in emp_id int

Goal: To permit an employee to view tickets raised by him/her



```
1   call TicketRaisedByOwn(7020);
```

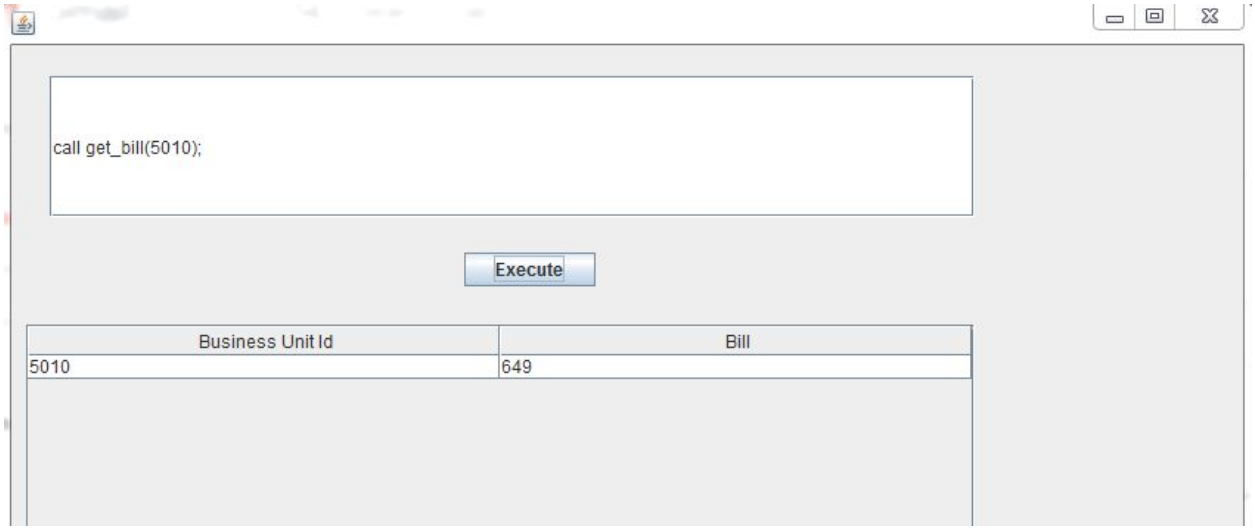
Result Grid | Filter Rows: | Export: | Wrap Cell Content:

ticket_id	ticket_type	end_user_email	severity	priority	opened_date	closed_date	opened_by	status
1000	Incident	elit@elit.co.uk	Low	Change Request	2010-09-04 00:00:00	2010-09-18 00:00:00	7020	Resolved
1019	Incident	Etiam.imperdiet.dictum@erat.edu	High	Work Request	2018-10-13 00:00:00	2017-09-10 00:00:00	7020	Work In Progress
1024	Incident	Nullam.feugiat.placerat@mollis.co.uk	Low	Work Request	2018-10-12 00:00:00	0000-00-00 00:00:00	7020	Work In Progress
1025	Problem	nisi.Aenean@Proinnon.com	Low	Work Request	2018-10-11 00:00:00	0000-00-00 00:00:00	7020	Work In Progress
1026	Change Request	pede.ultrices@erosNam.ca	High	Change Request	2017-09-15 00:00:00	2017-09-15 00:00:00	7020	Resolved
1029	Work Request	arcu.Morbi.sit@eratVivamusnisi.com	Medium	Work Request	2018-10-09 00:00:00	0000-00-00 00:00:00	7020	Work In Progress

12) *getBill*

Parameters:n business_unit_id int

Goal: To permit CEO of the IT organisation see the bills of various business units



Trigger:

1)*after_ticket_table_insert* : Trigger type: *after insert* on table *Ticket*

Goal: this trigger tracks the creation of new ticket. If a new Ticket is inserted in the Ticket database then the corresponding entry is inserted in the TicketTransactionActivity table with the activity as *Ticket Created.*

2)*after_ticket_table_update* : Trigger type: *after update* on Table *Ticket*.

Goal: this trigger tracks the updation of ticket already existing in the database. If there is assignment change from one person to another then the corresponding entry will be made in TicketTransactionActivity . The TicketTransactionActivity will bear activity as *Assignment Change* and where past_value column refers to the person who previously held the ticket while new_value column will store the information of the new assignee.
Similarly whenever there is a change in status of the ticket a new entry will be generated in the TicketTransactionActivity table which will that will bear the activity value as *Status Change* while the past_value will bear the previous status and new_value will contain the new status of the ticket.

```
show triggers;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Trigger	Event	Table	Statement	Timing	Created	sql_mode
after_ticket_table_insert	INSERT	Ticket	begin insert into TicketActivityLog (ticket_id,ac...	AFTER	2018-12-01 22:12:06.11	NO_ENGINE_SUBSTITUTION
after_ticket_table_update	UPDATE	Ticket	begin if !(NEW.assigned_to <=> OLD.assi...	BEFORE	2018-12-01 22:12:25.68	NO_ENGINE_SUBSTITUTION

Events:

1)*auto_assignment_ticket_event* : Type: RECURRING

Goal: The goal of this trigger is to call the stored procedure that auto assigns the unattended tickets to an employee's superior. If the ticket is in 'Open' state for more than 48 hours and no action has been taken on it it will automatically assigned to the an employee's superior. This event is scheduled at every 1 minute indefinitely for the purpose of demonstration. In real life scenario it could be around every 1 hour.

2)*auto_closure_ticket_event* : type: RECURRING

Goal: The goal of this trigger is to call the stored procedure that auto closes the ticket if the ticket is in resolved state and has been in the same state since past 48 hours.This event is scheduled at every 1 minute indefinitely for the purpose of demonstration. In real life scenario it could be around every 1 hour.

```
show events;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Db	Name	Definer	Time zone	Type	Execute at	Interval value	Interval field	Starts
TicketingToolTest	auto_assignment_ticket_event	admin@%	UTC	RECURRING	NULL	1	MINUTE	2018-12-01 14:21:51
TicketingToolTest	auto_closure_ticket_event	admin@%	UTC	RECURRING	NULL	1	MINUTE	2018-12-01 12:46:25

When the previous stored procedure *autoClosureTicket* was executed the following entries were generated in the database as Ticket table was updated:

```
109 •  select * from TicketActivityLog order by time_stamp desc;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap

ticket_id	activity	past_value	new_value	time_stamp
1003	Status Change	Resolved	Closed	2018-12-02 00:10:25

When the previous stored procedure *autoAssignmentTicket* was called the following entry was generated in the TicketTransaction Activity Table

```
109 •  select * from TicketActivityLog order by time_stamp desc;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Ce

ticket_id	activity	past_value	new_value	time_stamp
1000	Assignment Change	7019	7009	2018-12-02 00:04:51