Project Report On

TICKET MANAGEMENT SYSTEM

Professor:

James McCart

Prepared By:

Mahima Thakur (U245418885)
Mayank Yadav
(U94319493)
Sai Vamsi Neeluri (U20653235)
Swati Nln
(U69103767)
Venkat Chander Gonguluri
(U44484940)

April 2018

Muma College of Business
University of South Florida

INDEX

FIGURES

TABLES

INTRODUCTION

A Ticketing System streamlines, centralizes and manages all your tickets. Take email requests and transform them into service tickets automatically. From the initial ticket request and creation to resolution and closure ticketing system software helps you through every stage.

A Ticketing System allows you to automate ticket assignment, routing, and escalation to the right agent at the right time. Save time and manual effort for the help desk team, and improve help desk agent productivity. With the help of Web-based portal to log tickets, and an intuitive console for technicians to manage tickets. A ticketing system software enables you to:

- Automate ticket assignment, routing and escalation to the right agent, at the right time
- Centralize Ticket Management – from request creation to resolution
- Save time and resources on manual and repetitive tasks
- Track and monitor help desk and technician performance in real time
- Improve operational efficiency of customer service
- Realize higher levels of customer satisfaction

A Ticketing System centralizes the channel for receiving service requests via an interactive Web portal. Teams can benefit from avoiding the use of disparate sources such as chat, phone, email and in-person discussions for customer communication. Once the request is logged in the Ticketing System, pros communicate back to the end-user with acknowledgement and update ticket fulfillment progress including technician assignment and expected request fulfillment date/time. Ticketing software also converts inbound service requests via email into tickets in the Ticketing System.

ASSUMPTIONS

- It is assumed that every Client will login using their Unique Email Ids.
- Staff table is kept separate from the Employee table.
- A Staff can only block a Client.
- The client while registering and updating his/her profile is entering all details correctly like street address, city, state, country, zip code, etc.
- If the Ticket is created by Client, the default Priority is Normal.
- Only Client can interact with each other in Forum.
- A Client can reply more than once in a post.
- Only Staff can set the field “isDeleted” to true in any tables.
- In Case, the company want to investigate any deleted record, they will have to contact Database Admin.

NOTE: The operation on database were done on separate connections. For performance tuning we populated each table with 6000-60000 rows to have better performance for large database. We did not use performance tuning database connection for rest of the project, therefore you might observe some statistical changes. We also implemented a Module, in which client will receive a mail of reply posted by staff. We implemented it in ASP.NET. You may find interesting queries spread across different sections. We have introduced different topics as they are required, for example, the Queries and Data Visualisation section are concatenated together for better visualisation.

SYSTEM DESIGN

Entity Relationship Diagram

There are specific functional areas in this database. The complete ERD is displayed in Figure 1. All the functional areas are discussed in detail in the database decomposition section.

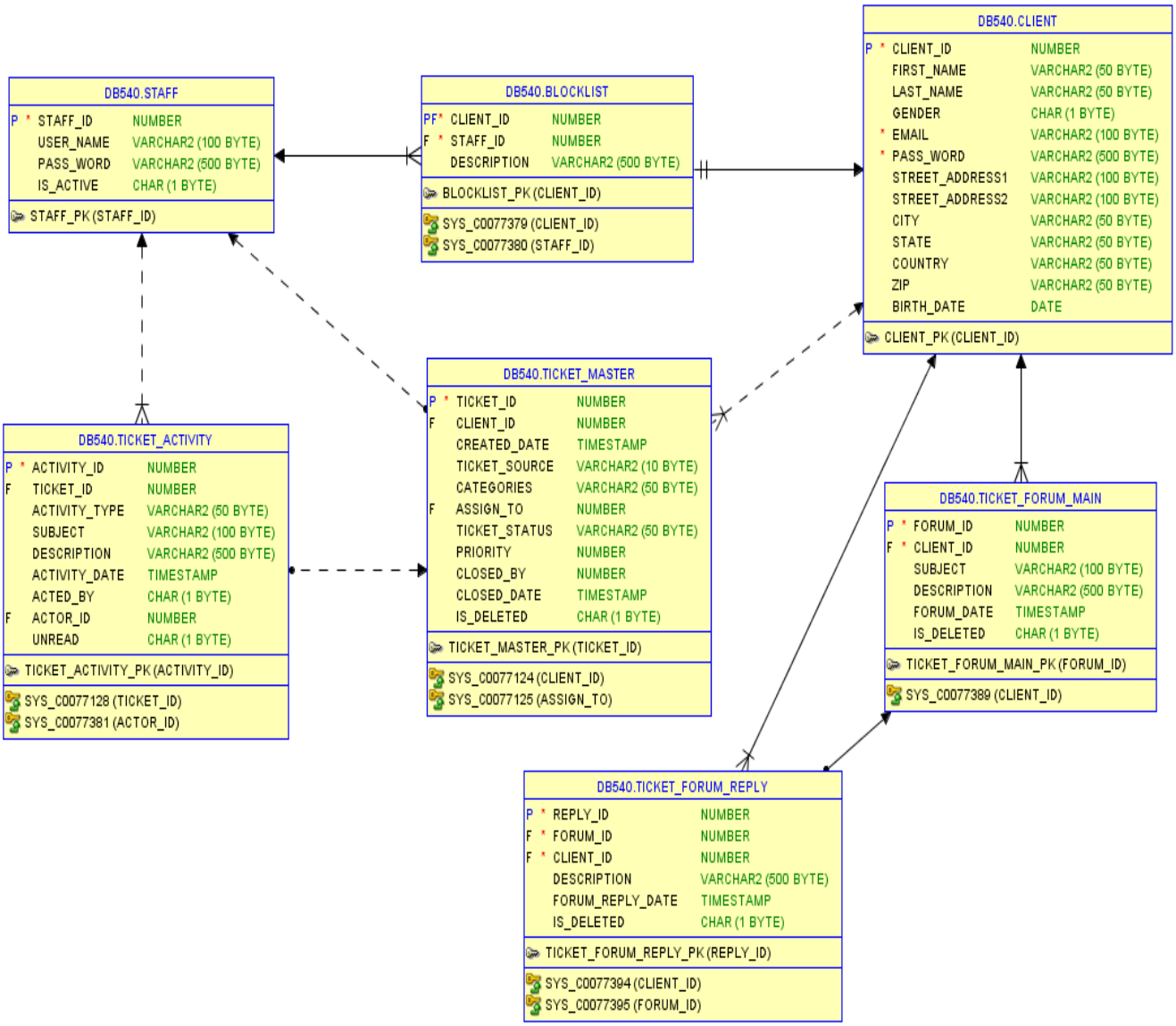


Figure 1 : ER Diagram

Functional Decomposition Diagram

The Movies database consists of 7 subsets (Figure 2). These functional areas include Staff, Client, Ticket Activity, Ticket Master, Ticket Activity, Ticket forum reply and Ticket main forum and Blocklist. Each of them represent a key area. These areas can be possibly distinguished into separate databases in future according to the requirements.

An effective system has the ability to classify support tickets in order to enable reporting on issue types and skills required on the basis of how the tickets are routed. Such an application can be useful as it is less time consuming than manual ticket management practices. This opportunity for improved curation accuracy is vital in evolving businesses.

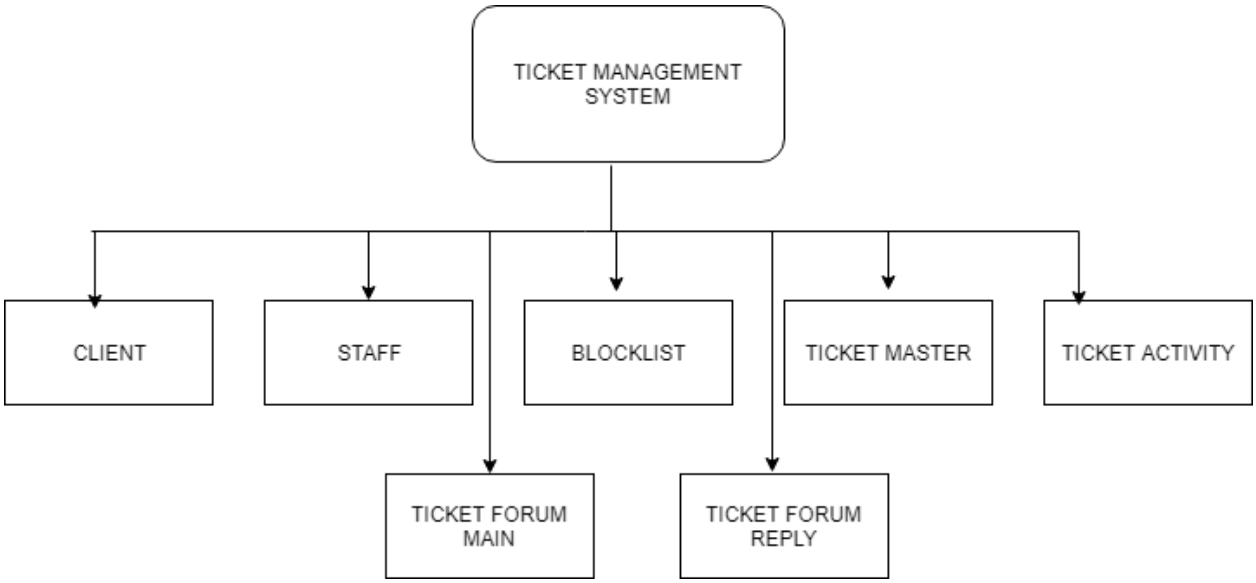


Figure 2 : Database Decomposition

Data Dictionary

Staff

	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	STAFF_ID	NUMBER	No	(null)	1	(null)
2	USER_NAME	VARCHAR2(100 BYTE)	Yes	(null)	2	(null)
3	PASS_WORD	VARCHAR2(500 BYTE)	Yes	(null)	3	(null)
4	IS_ACTIVE	CHAR(1 BYTE)	Yes	(null)	4	(null)

Table 1 : Staff Table

The staff table is kept separate from the Employee table, considering only those columns relevant for this system. “STAFF_ID” is the primary key for Staff Table. It works in an auto-increment fashion. Created a sequence named “STAFF_SEQ” and will be called every time while inserting a new record. This table is responsible for storing the username of the staff members. Instead of deleting a staff record, the “IS_ACTIVE” field will be set to “N” which stands for No, otherwise will be set to “Y” for Yes.

Client

	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	CLIENT_ID	NUMBER	No	(null)	1	(null)
2	FIRST_NAME	VARCHAR2(50 BYTE)	Yes	(null)	2	(null)
3	LAST_NAME	VARCHAR2(50 BYTE)	Yes	(null)	3	(null)
4	GENDER	CHAR(1 BYTE)	Yes	(null)	4	(null)
5	EMAIL	VARCHAR2(100 BYTE)	No	(null)	5	(null)
6	PASS_WORD	VARCHAR2(500 BYTE)	No	(null)	6	(null)
7	STREET_ADDRESS1	VARCHAR2(100 BYTE)	Yes	(null)	7	(null)
8	STREET_ADDRESS2	VARCHAR2(100 BYTE)	Yes	(null)	8	(null)
9	CITY	VARCHAR2(50 BYTE)	Yes	(null)	9	(null)
10	STATE	VARCHAR2(50 BYTE)	Yes	(null)	10	(null)
11	COUNTRY	VARCHAR2(50 BYTE)	Yes	(null)	11	(null)
12	ZIP	VARCHAR2(50 BYTE)	Yes	(null)	12	(null)
13	BIRTH_DATE	DATE	Yes	(null)	13	(null)

Table 2 : Client Table

This table is responsible for storing Client details. “Client_ID” is the primary key for Client Table. It also works in an auto-increment fashion. Created a sequence named “CLIENT_SEQ” and will be called every time while inserting a new record. Gender is set to Char data type with “M” standing for Male and “F” for Female.It is assumed that client will enter the address correctly. The email should be unique for every record and is username while signing in the customer. Client_ID, Email and Password are not null values. Created “check_client_birth_date” trigger to check if client’s date of birth is later than Jan 1, 1900 and earlier than today. Also there is function named “sha”, responsible for converting the password to SHA hash of 512 or 256 bits. A scheduler is created for getting reminder for client’s birthdays.

Blocklist

	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	CLIENT_ID	NUMBER	No	(null)	1	(null)
2	STAFF_ID	NUMBER	No	(null)	2	(null)
3	DESCRIPTION	VARCHAR2(500 BYTE)	Yes	(null)	3	(null)

Table 3 : Blocklist Table

"CLIENT_ID" and STAFF_ID are foreign keys from Client and Staff table respectively. This table will keep track of clients who are not allowed to create a ticket. STAFF_ID is responsible for knowing which staff member has blocked a particular client and "DESCRIPTION" is for describing the reason as to why the client was blocked.

Ticket_Master

	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	TICKET_ID	NUMBER	No	(null)	1	(null)
2	CLIENT_ID	NUMBER	Yes	(null)	2	(null)
3	CREATED_DATE	TIMESTAMP(6)	Yes	(null)	3	(null)
4	TICKET_SOURCE	VARCHAR2(10 BYTE)	Yes	(null)	4	(null)
5	CATEGORIES	VARCHAR2(50 BYTE)	Yes	(null)	5	(null)
6	ASSIGN_TO	NUMBER	Yes	(null)	6	(null)
7	TICKET_STATUS	VARCHAR2(50 BYTE)	Yes	(null)	7	(null)
8	PRIORITY	NUMBER	Yes	(null)	8	(null)
9	CLOSED_BY	NUMBER	Yes	(null)	9	(null)
10	CLOSED_DATE	TIMESTAMP(6)	Yes	(null)	10	(null)
11	IS_DELETED	CHAR(1 BYTE)	Yes	(null)	11	(null)

Table 4 : Ticket Master Table

"TICKET_ID" is the primary key of "Ticket_Master" table. Created a sequence named "TICKET_MASTER_SEQ" and will be called every time while inserting new record. "TICKET_SOURCE" has three values: W for Web, E for Email and P for Phone Call. "ASSIGN_TO" stores the "STAFF_ID" of the user who has been assigned the particular ticket to solve. If the ticket is created by emailing them then the value is set to null and later on it can be updated. "TICKET_STATUS" has following values: New, Open, Awaiting Customer Response, Closed. The default value is New. Priority has five values: 0 - Low, 1- Normal, 2- High, 3 – Emergency. This system does not delete any record the database, we just set the "IS_DELETED" deleted to true or yes or 1, therefore in future if there is an inspection, we can go through all records.

Ticket_Activity

	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	ACTIVITY_ID	NUMBER	No	(null)	1	(null)
2	TICKET_ID	NUMBER	Yes	(null)	2	(null)
3	ACTIVITY_TYPE	VARCHAR2(50 BYTE)	Yes	(null)	3	(null)
4	SUBJECT	VARCHAR2(100 BYTE)	Yes	(null)	4	(null)
5	DESCRIPTION	VARCHAR2(500 BYTE)	Yes	(null)	5	(null)
6	ACTIVITY_DATE	TIMESTAMP(6)	Yes	(null)	6	(null)
7	ACTED_BY	CHAR(1 BYTE)	Yes	(null)	7	(null)
8	ACTOR_ID	NUMBER	Yes	(null)	8	(null)
9	UNREAD	CHAR(1 BYTE)	Yes	(null)	9	(null)

Table 5 : Ticket Activity Table

This table will store any activity done on Ticket. "ACTIVITY_ID" is the primary key of table "TicketActivity" and is auto incremental. Created a sequence named "TICKET_ACTIVITY_SEQ". It will be called every time while inserting new record. "TICKET_ID" is a Foreign Key from "Ticket_Master". "ACTIVITY_TYPE" can have following value: Created, Updated, Deleted. "ACTED_BY" will have the "Staff_ID" but if Ticket is created via email then it will be null and "ACTED_BY" will be End User. "UNREAD" is a Boolean value set to keep track of unread activities.

Ticket_Forum_Main

	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	FORUM_ID	NUMBER	No	(null)	1	(null)
2	CLIENT_ID	NUMBER	No	(null)	2	(null)
3	SUBJECT	VARCHAR2(100 BYTE)	Yes	(null)	3	(null)
4	DESCRIPTION	VARCHAR2(500 BYTE)	Yes	(null)	4	(null)
5	FORUM_DATE	TIMESTAMP(6)	Yes	(null)	5	(null)
6	IS_DELETED	CHAR(1 BYTE)	Yes	(null)	6	(null)

Table 6 : Ticket Forum Main Table

FORUM_ID is the primary key for Ticket_Forum_Main Table. It is also implemented in an auto increment fashion. Created a sequence named "TICKET_FORUM_MAIN_SEQ" and will be called every time while inserting new record. This table stores records for forum. There is a forum on which clients can create a forum and have other clients post reply to their problem. CLIENT_ID is foreign key to the Client.

Ticket_Forum_Reply

	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	REPLY_ID	NUMBER	No	(null)	1	(null)
2	FORUM_ID	NUMBER	No	(null)	2	(null)
3	CLIENT_ID	NUMBER	No	(null)	3	(null)
4	DESCRIPTION	VARCHAR2(500 BYTE)	Yes	(null)	4	(null)
5	FORUM_REPLY_DATE	TIMESTAMP(6)	Yes	(null)	5	(null)
6	IS_DELETED	CHAR(1 BYTE)	Yes	(null)	6	(null)

Table 7 : Ticket Forum Reply Table

REPLY_ID is the primary key for Ticket_Forum_Reply table. It is also implemented in an auto increment fashion. Created a sequence named "TICKET_FORUM_REPLY_SEQ" and will be called every time while inserting new record. FORUM_ID and CLIENT_ID are foreign keys to Forum Table and Client table respectively.

SQL Developer

Component	Version
Java(TM) Platform	1.8.0_151
Oracle IDE	17.4.0.355.2349
Versioning Support	17.4.0.355.2349

Table 8 : SQL Developer Details

Properties

Name	Value
awt.toolkit	sun.awt.windows.WToolkit
copyright.year.end	2017
copyright.year.start	2005
eclipse.home.location	file:/E:/Education/Graduate/Sem 2/ADM/sqldeveloper/
eclipse.parsers.setTCCL	false
excluded.modules	org.eclipse.osgi
felix.log.level	4
file.encoding	Cp1252
file.encoding.pkg	sun.io

Name	Value
ide.bundle.search.path	E:\Education\Graduate\Sem 2\ADM\sqldeveloper\dropins
ide.cfu.class.path	E:\Education\Graduate\Sem 2\ADM\sqldeveloper\ide\macros/../../modules/oracle.bali.share/share.jar,file:../../modules/oracle.bali.jewt/jewt4
ide.cluster.dirs	/Education/Graduate/Sem 2/ADM/sqldeveloper/netbeans/fcpbridge/:/Education/Graduate/Sem 2/ADM/sqldeveloper/netbeans/ide/:/Education
ide.conf	E:\Education\Graduate\Sem 2\ADM\sqldeveloper\sqldeveloper\bin\sqldeveloper.conf
ide.config_pathname	E:\Education\Graduate\Sem 2\ADM\sqldeveloper\sqldeveloper\bin\sqldeveloper.conf
ide.debugbuild	false
ide.devbuild	false
ide.diagnostics.dir	C:\Users\MAYANK YADAV\AppData\Roaming\SQL Developer\system17.4.0.355.2349\.diagnostics\
ide.extension.search.path	sqldeveloper/extensions:jdev/extensions:ide/extensions
ide.feedback-server	ide.us.oracle.com
ide.firstrun	true
ide.java.maxversion	9.1
ide.java.minversion	1.8.0_40
ide.launcherProcessId	11476
ide.mw.relative.home	.
ide.pref.dir	C:\Users\MAYANK YADAV\AppData\Roaming\SQL Developer
ide.pref.dir.base	C:\Users\MAYANK YADAV\AppData\Roaming
ide.product	oracle.sqldeveloper
ide.RectangularSplashScreen	true
ide.shell.enableFileTypeAssociation	E:\Education\Graduate\Sem 2\ADM\sqldeveloper\sqldeveloper\bin\sqldeveloper64W.exe
ide.splash.screen	splash.gif
ide.startingArg0	E:\Education\Graduate\Sem 2\ADM\sqldeveloper\sqldeveloper\bin\sqldeveloper64W.exe
ide.startingcwd	E:\Education\Graduate\Sem 2\ADM\sqldeveloper\sqldeveloper\bin
ide.system.dir	C:\Users\MAYANK YADAV\AppData\Roaming\SQL Developer\system17.4.0.355.2349\
ide.update.usage.servers	https://www.oracle.com/webfolder/technetwork/sqldeveloper/usage.xml
ide.user.dir	C:\Users\MAYANK YADAV\AppData\Roaming\SQL Developer

Name	Value
ide.user.dir.var	IDE_USER_DIR
ide.vcs.noapplications	true
ide.work.dir	C:\Users\MAYANK YADAV\Documents\SQL Developer
ide.work.dir.base	C:\Users\MAYANK YADAV\Documents
ilog.propagatesPropertyEditors	false
InstanceDataObject.current.file	
java.awt.graphicsenv	sun.awt.Win32GraphicsEnvironment
java.awt.printerjob	sun.awt.windows.WPrinterJob
java.class.path	E:\Education\Graduate\Sem 2\ADM\sqldeveloper\jde\lib\jde-boot.jar;E:\Education\Graduate\Sem 2\ADM\sqldeveloper\netbeans\platform\lib\b
java.class.version	52.0
java.endorsed.dirs	E:\Education\Graduate\Sem 2\ADM\sqldeveloper\jdk\jre\lib\endorsed
java.ext.dirs	E:\Education\Graduate\Sem 2\ADM\sqldeveloper\jdk\jre\lib\ext;C:\WINDOWS\Sun\Java\lib\ext
java.home	E:\Education\Graduate\Sem 2\ADM\sqldeveloper\jdk\jre
java.io.tmpdir	C:\Users\MAYANK~1\AppData\Local\Temp\
java.library.path	E:\Education\Graduate\Sem 2\ADM\sqldeveloper\sqldeveloper\bin;C:\WINDOWS\Sun\Java\bin;C:\WINDOWS\system32;C:\WINDOWS;C:\Pro
java.memory.heap.init	134217728
java.memory.heap.max	746061824
java.memory.heap.used	169263792
java.memory.nonheap.init	2555904
java.memory.nonheap.max	-1
java.memory.nonheap.used	248002376
java.naming.factory.initial	oracle.javatools.jndi.LocalInitialContextFactory
java.runtime.name	Java(TM) SE Runtime Environment
java.runtime.version	1.8.0_151-b12
java.specification.name	Java Platform API Specification
java.specification.vendor	Oracle Corporation
java.specification.version	1.8
java.util.logging.config.file	logging.conf
java.vendor	Oracle Corporation
java.vendor.url	http://java.oracle.com/
java.vendor.url.bug	http://bugreport.sun.com/bugreport/
java.version	1.8.0_151
java.vm.info	mixed mode
java.vm.name	Java HotSpot(TM) 64-Bit Server VM
java.vm.specification.name	Java Virtual Machine Specification
java.vm.specification.vendor	Oracle Corporation
java.vm.specification.version	1.8
java.vm.vendor	Oracle Corporation
java.vm.version	25.151-b12
javax.xml.parsers.DocumentBuilderFactory	oracle.xml.jaxp.JXDocumentBuilderFactory
javax.xml.parsers.SAXParserFactory	oracle.xml.jaxp.JXSAXParserFactory
javax.xml.stream.util.XMLEventAllocator	oracle.ideimpl.xml.stream.XMLEventAllocatorImpl
javax.xml.stream.XMLInputFactory	com.ctc.wstx.stax.WstxInputFactory
javax.xml.transform.TransformerFactory	oracle.ide.xml.switchable.SwitchableTransformerFactory
jdbc.library	E:\Education\Graduate\Sem 2\ADM\sqldeveloper\jdbc\lib\ojdbc8.jar
jdeveloper.system_http_proxy	DIRECT
jdk.home	E:\Education\Graduate\Sem 2\ADM\sqldeveloper\jdk\jre\..

Figure 3 : SQL Developer

DATABASES

Staff

Data

	STAFF_ID	USER_NAME	PASS_WORD	IS_ACTIVE
1	21	PETER	peter123	Y
2	22	MEG	meg123	Y
3	1	Admin	admin123	Y

Figure 4 : Staff - Data

Model

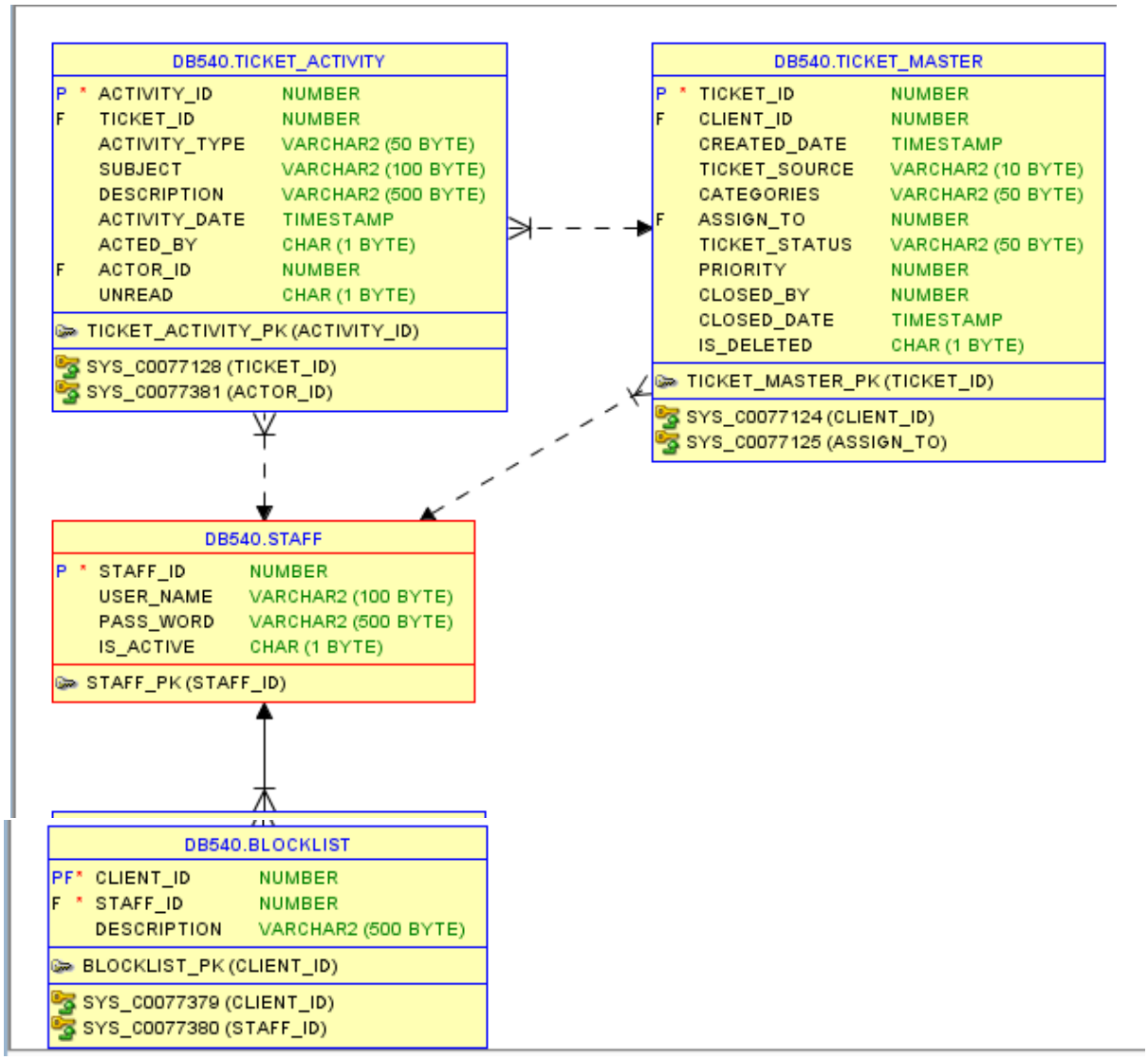


Figure 5 : Staff - Model

Constraints

	CONSTRAINT_NAME	CONSTRAINT_TYPE	SEARCH_CONDITION	R_OWNER	R_TABLE_NAME	R_CONSTRAINT_NAME	DELETE_RULE	STATUS	DEFERRABLE	VALIDATED
1	SYS_C0077116	Check	"STAFF_ID" IS NOT NULL	(null)	(null)	(null)	(null)	ENABLED	NOT DEFERRABLE	VALIDATED
2	SYS_C0077117	Primary_Key	(null)	(null)	(null)	(null)	(null)	ENABLED	NOT DEFERRABLE	VALIDATED

Figure 6 : Staff - Constraints

Client

Data

	CLIENT_ID	FIRST_NAME	LAST_NAME	GENDER	EMAIL	PASS_WORD	STREET_ADDRESS1	STREET_ADDRESS2	CITY	STATE	COUNTRY	ZIP	BIRTH_DAT
1	21	ABC	DEF	M	abc@abc.com	abcdef	(null)	(null)	(null)	(null)	(null)	(null)	(null)
2	43	TONY	STARK	M	stark@stark.com	stark@tony	(null)	(null)	(null)	(null)	(null)	(null)	26-MAR-96
3	44	TOM	JERRY	M	tom@tom.com	tom@jerry	(null)	(null)	(null)	(null)	(null)	(null)	25-APR-95
4	45	ROAD	RUNNER	M	road@road.com	road@runner	(null)	(null)	(null)	(null)	(null)	(null)	24-MAY-94
5	46	BOB	BUILDER	M	bob@bob.com	bob@builder	(null)	(null)	(null)	(null)	(null)	(null)	23-JUN-93
6	1	Mayank	Yadav	M	mayanky@mail.usf.edu	P@ssowrd123	(null)	(null)	(null)	(null)	(null)	(null)	(null)

Figure 7 : Client - Data

Model

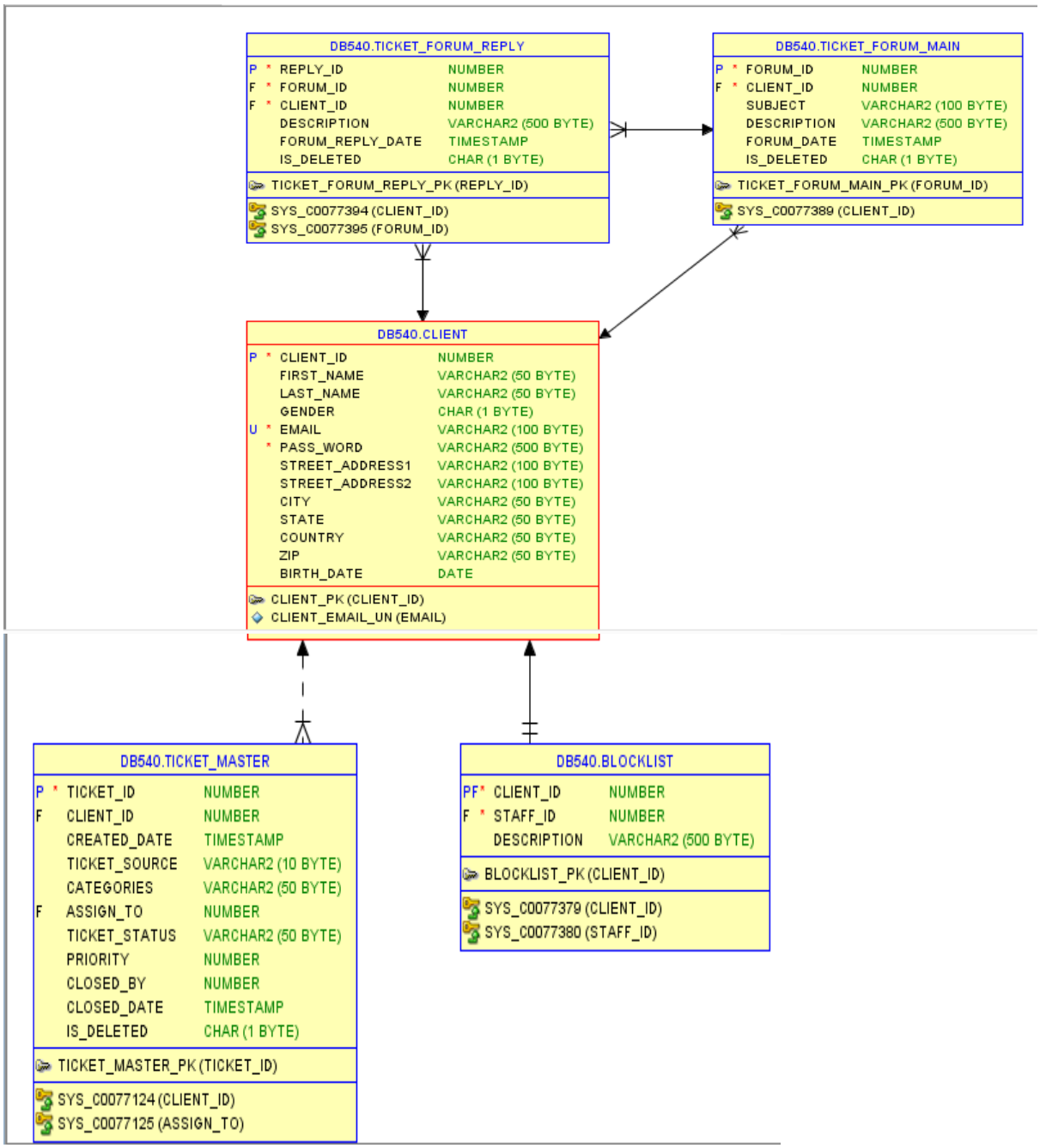


Figure 8 : Client - Model

Constraints

	CONSTRAINT_NAME	CONSTRAINT_TYPE	SEARCH_CONDITION	R_OWNER	R_TABLE_NAME	R_CONSTRAINT_NAME	DELETE_RULE	STATUS	DEFERRABLE	VALIDATED
1	SYS_C0077112	Check	"CLIENT_ID" IS NOT NULL	(null)	(null)	(null)	(null)	ENABLED	NOT DEFERRABLE	VALIDATED
2	SYS_C0077113	Check	"EMAIL" IS NOT NULL	(null)	(null)	(null)	(null)	ENABLED	NOT DEFERRABLE	VALIDATED
3	SYS_C0077114	Check	"PASS_WORD" IS NOT NULL	(null)	(null)	(null)	(null)	ENABLED	NOT DEFERRABLE	VALIDATED
4	SYS_C0077115	Primary_Key	(null)	(null)	(null)	(null)	(null)	ENABLED	NOT DEFERRABLE	VALIDATED
5	SYS_C0078200	Unique	(null)	(null)	(null)	(null)	(null)	ENABLED	NOT DEFERRABLE	VALIDATED

Figure 9 : Client - Constraints

Ticket Master

Data

	TICKET_ID	CLIENT_ID	CREATED_DATE	TICKET_SOURCE	CATEGORIES	ASSIGN_TO	TICKET_STATUS	PRIORITY	CLOSED_BY	CLOSED_DATE	IS_DELETED
1	21	43	07-APR-18 10.03.47.847000000 AM	Web	Complain	1	Waiting	2	(null)	(null)	N
2	22	44	07-APR-18 10.03.50.327000000 AM	Web	Request	21	Resolved	3	(null)	(null)	N
3	23	45	07-APR-18 10.03.52.667000000 AM	Web	Complain	(null)	New	4	(null)	(null)	N
4	1	1	25-MAR-18 09.35.22.620000000 PM	Web	Request	1	New	1	(null)	(null)	N

Figure 10 : Ticket Master - Data

Model

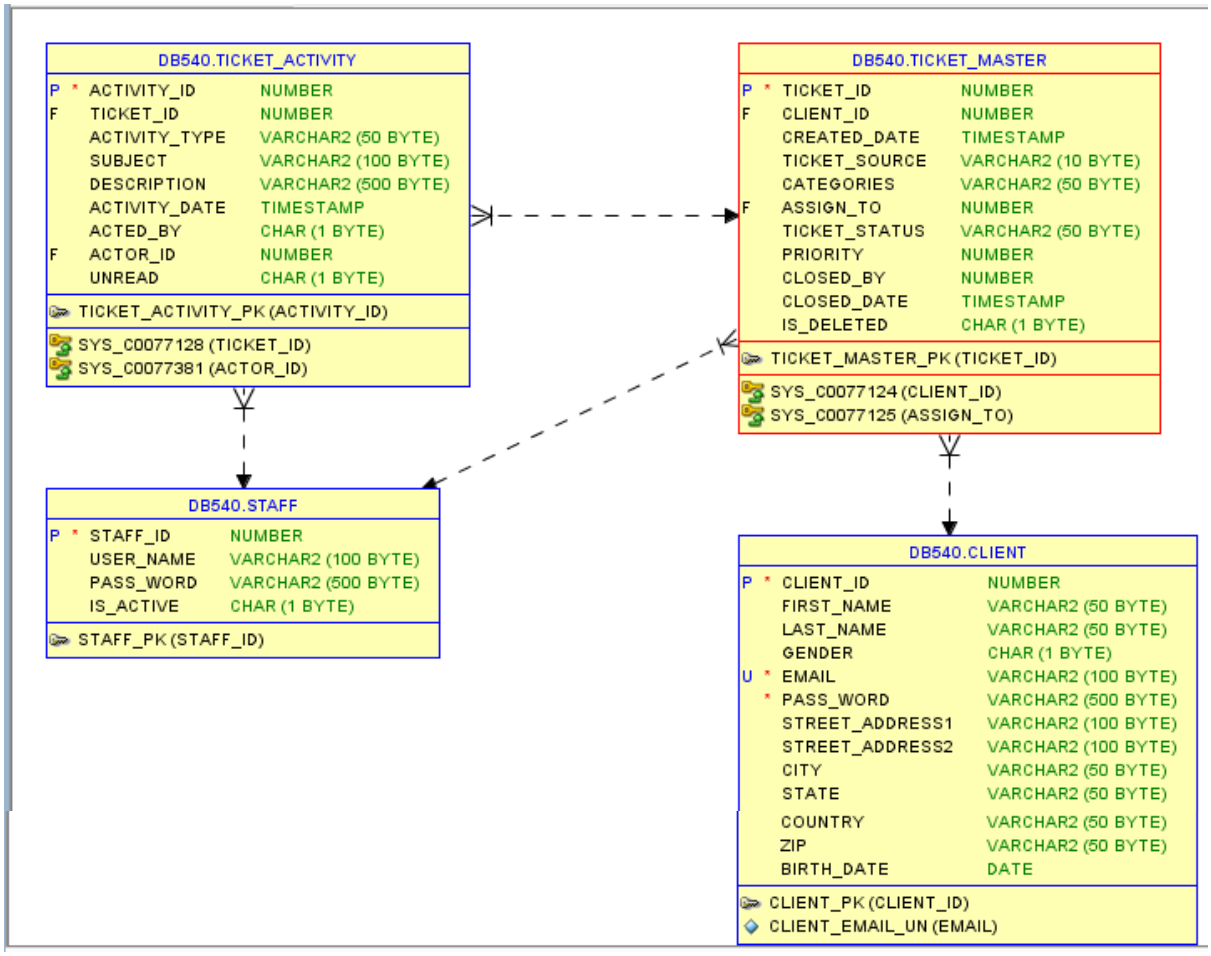


Figure 11 : Ticket Master Model

Constraints

	CONSTRAINT_NAME	CONSTRAINT_TYPE	SEARCH_CONDITION	R_OWNER	R_TABLE_NAME	R_CONSTRAINT_NAME	DELETE_RULE	STATUS	DEFERRABLE	VALIDATED
1	SYS_C0077122	Check	"TICKET_ID" IS NOT NULL	(null)	(null)	(null)	(null)	ENABLED	NOT DEFERRABLE	VALIDATED
2	SYS_C0077123	Primary_Key	(null)	(null)	(null)	(null)	(null)	ENABLED	NOT DEFERRABLE	VALIDATED
3	SYS_C0077124	Foreign_Key	(null)	DB540	CLIENT	SYS_C0077115	NO ACTION	ENABLED	NOT DEFERRABLE	VALIDATED
4	SYS_C0077125	Foreign_Key	(null)	DB540	STAFF	SYS_C0077117	NO ACTION	ENABLED	NOT DEFERRABLE	VALIDATED

Figure 12 : Ticket Master - Constraints

Ticket Activity

Data

	ACTIVITY_ID	TICKET_ID	ACTIVITY_TYPE	SUBJECT	DESCRIPTION	ACTIVITY_DATE	ACTED_BY	ACTOR_ID	UNREAD
1	21	21	Create	Complian about a...	I would like to Complian...	07-APR-18 10.08.46.275000000 AM	C	(null)	N
2	22	21	Assign	Staff has been a...	Admin (1) has been assig...	07-APR-18 10.18.04.008000000 AM	S	1	N
3	23	21	Reply	Reply to Complia...	It was a company fault. ...	07-APR-18 10.18.16.940000000 AM	S	1	N
4	24	21	Priority	Priority Changed	Your ticket priority has...	07-APR-18 10.18.19.171000000 AM	S	1	N
5	25	21	Ticket Status	Ticket Status ha...	Ticket has been changed ...	07-APR-18 10.18.25.442000000 AM	S	1	N
6	26	22	Assign	Staff has been a...	Peter (21) has been assi...	07-APR-18 10.25.24.833000000 AM	S	21	N
7	27	22	Reply	Reply to Request...	We have stocked this pro...	07-APR-18 10.25.30.215000000 AM	S	21	N
8	28	22	Priority	Priority Changed	Your ticket priority has...	07-APR-18 10.25.32.181000000 AM	S	21	N
9	29	22	Ticket Status	Ticket Status ha...	Ticket has been changed ...	07-APR-18 10.25.36.034000000 AM	S	21	N
10	30	22	Customer	Customer Replied	Thank You	07-APR-18 10.25.41.946000000 AM	C	(null)	N
11	31	22	Ticket Status	Ticket Status ha...	Ticket has been changed ...	07-APR-18 10.25.43.990000000 AM	S	21	N
12	32	22	Create	Request for prod...	I would like to Request ...	07-APR-18 10.30.24.666000000 AM	C	(null)	N
13	33	23	Create	Complain about s...	I would like to Complain...	07-APR-18 10.30.30.703000000 AM	C	(null)	N
14	1	1	Create	Request for New ...	I would like to request ...	25-MAR-18 09.47.33.466000000 PM	C	1	N

Figure 13 : Ticket Activity - Data

Model

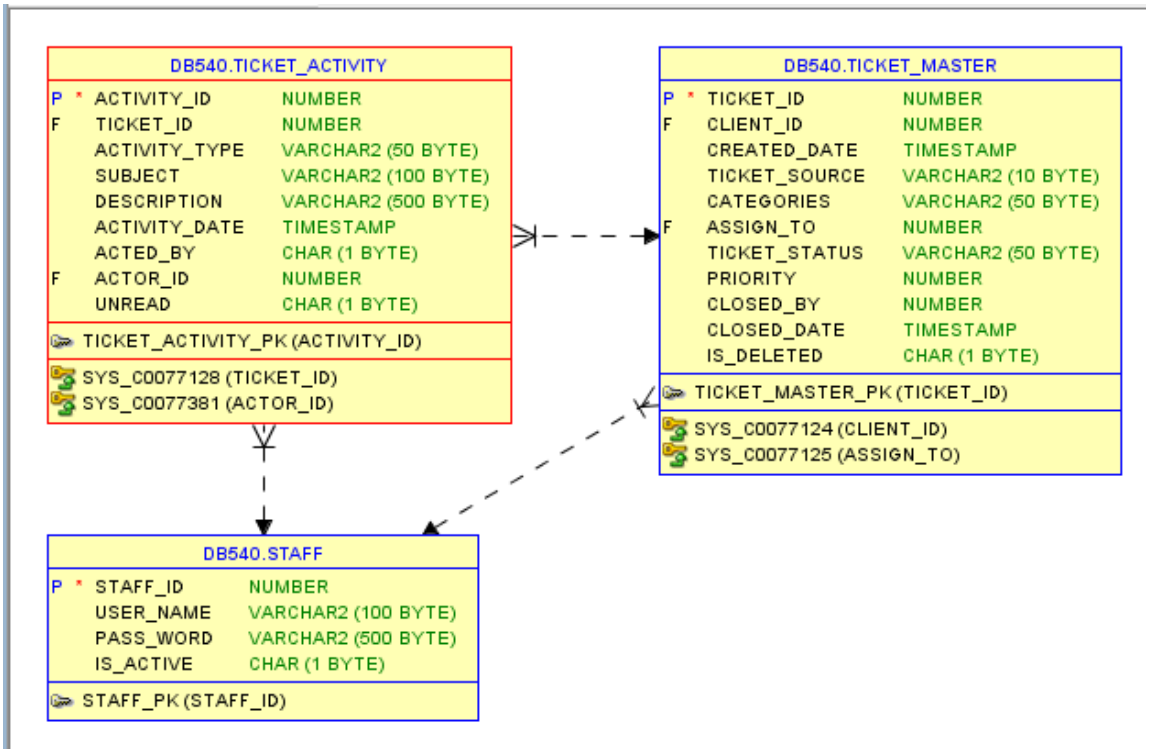


Figure 14 : Ticket Activity - Model

Constraints

	CONSTRAINT_NAME	CONSTRAINT_TYPE	SEARCH_CONDITION	R_OWNER	R_TABLE_NAME	R_CONSTRAINT_NAME	DELETE_RULE	STATUS	DEFERRABLE	VALIDATED
1	SYS_C0077126	Check	"ACTIVITY_ID" IS NOT NULL	(null)	(null)	(null)	(null)	ENABLED	NOT DEFERRABLE	VALIDATED
2	SYS_C0077127	Primary_Key	(null)	(null)	(null)	(null)	(null)	ENABLED	NOT DEFERRABLE	VALIDATED
3	SYS_C0077128	Foreign_Key	(null)	DB540	TICKET_MASTER	SYS_C0077123	NO ACTION	ENABLED	NOT DEFERRABLE	VALIDATED
4	SYS_C0077381	Foreign_Key	(null)	DB540	STAFF	SYS_C0077117	NO ACTION	ENABLED	NOT DEFERRABLE	VALIDATED

Figure 15 : Ticket Activity - Constraints

Ticket Forum Main

Data

	FORUM_ID	CLIENT_ID	SUBJECT	DESCRIPTION	FORUM_DATE	IS_DELETED
1	3	43	REVIEW OF NIVEA DEO	CAN YOU GUYS GIVE ME REVIEW OF THE NIVEA DEO	07-APR-18 04.01.47.837000000 PM	N
2	4	44	BEST HAIR OIL	I NEED SOME recommendation FOR BEST HAIR OIL	07-APR-18 04.01.50.239000000 PM	N

Figure 16 : Ticket Forum Main - Data

Model

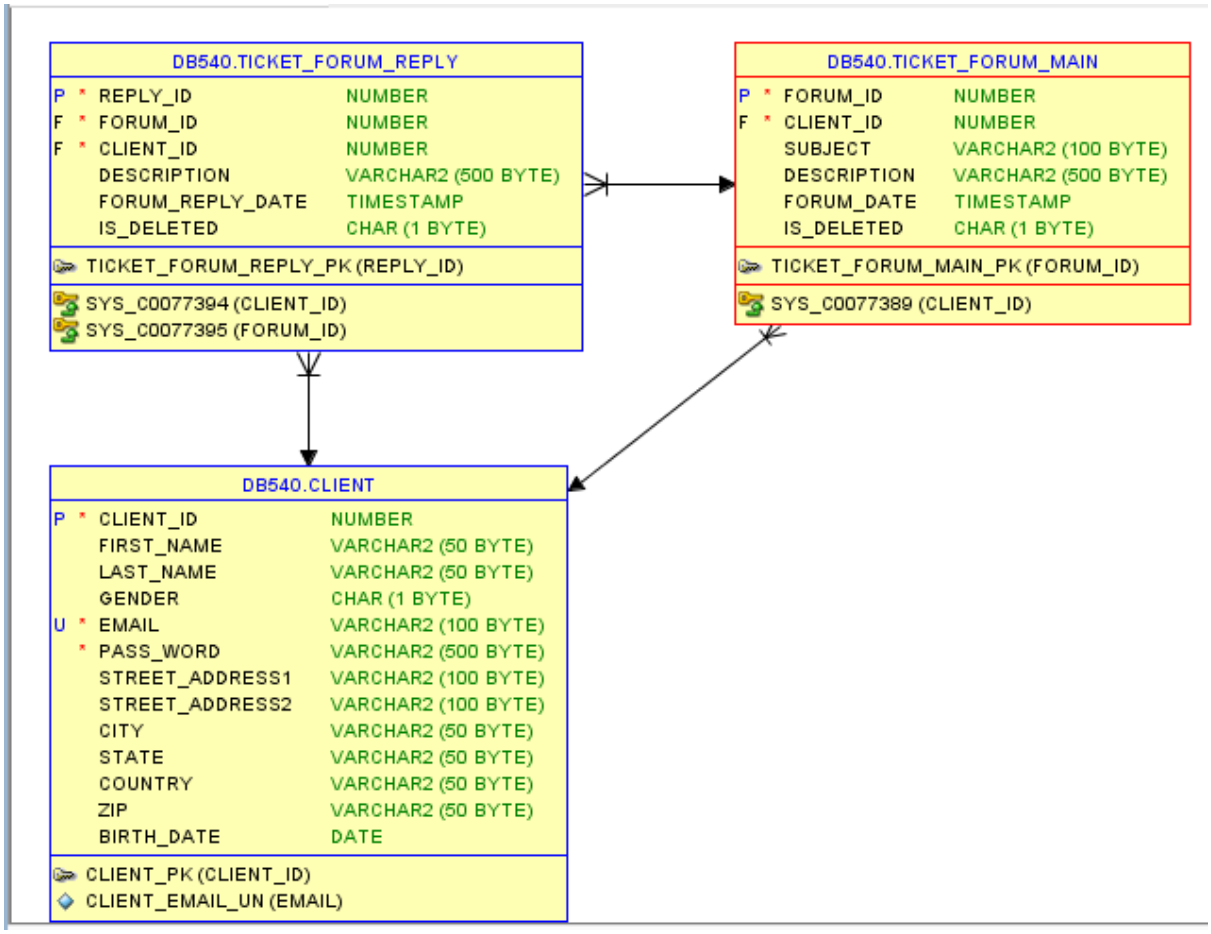


Figure 17 : Ticket Forum Main - Model

Constraints

	CONSTRAINT_NAME	CONSTRAINT_TYPE	SEARCH_CONDITION	R_OWNER	R_TABLE_NAME	R_CONSTRAINT_NAME	DELETE_RULE	STATUS	DEFERRABLE	VALIDATED
1	SYS_C0077386	Check	"FORUM_ID" IS NOT NULL	(null)	(null)	(null)	(null)	ENABLED	NOT DEFERRABLE	VALIDATED
2	SYS_C0077387	Check	"CLIENT_ID" IS NOT NULL	(null)	(null)	(null)	(null)	ENABLED	NOT DEFERRABLE	VALIDATED
3	SYS_C0077388	Primary_Key	(null)	(null)	(null)	(null)	(null)	ENABLED	NOT DEFERRABLE	VALIDATED
4	SYS_C0077389	Foreign_Key	(null)	DB540	CLIENT	SYS_C0077115	NO ACTION	ENABLED	NOT DEFERRABLE	VALIDATED

Figure 18 : Ticket Forum Main - Constraints

Ticket Forum Reply

Data

	REPLY_ID	FORUM_ID	CLIENT_ID	DESCRIPTION	FORUM_REPLY_DATE	IS_DELETED
1	2	3	45	CAN YOU GUYS GIVE ME REVIEW OF THE NIVEA DEO	07-APR-18 04.04.56.675000000 PM	N
2	3	4	46	I NEED SOME recommendation FOR BEST HAIR OIL	07-APR-18 04.04.58.578000000 PM	N
3	4	4	45	THIS IS GREAT	07-APR-18 04.54.13.520000000 PM	N
4	5	3	46	COCONUJT AND PARASHUT	07-APR-18 04.54.16.312000000 PM	N

Figure 19 : Ticket Forum Reply - Data

Model

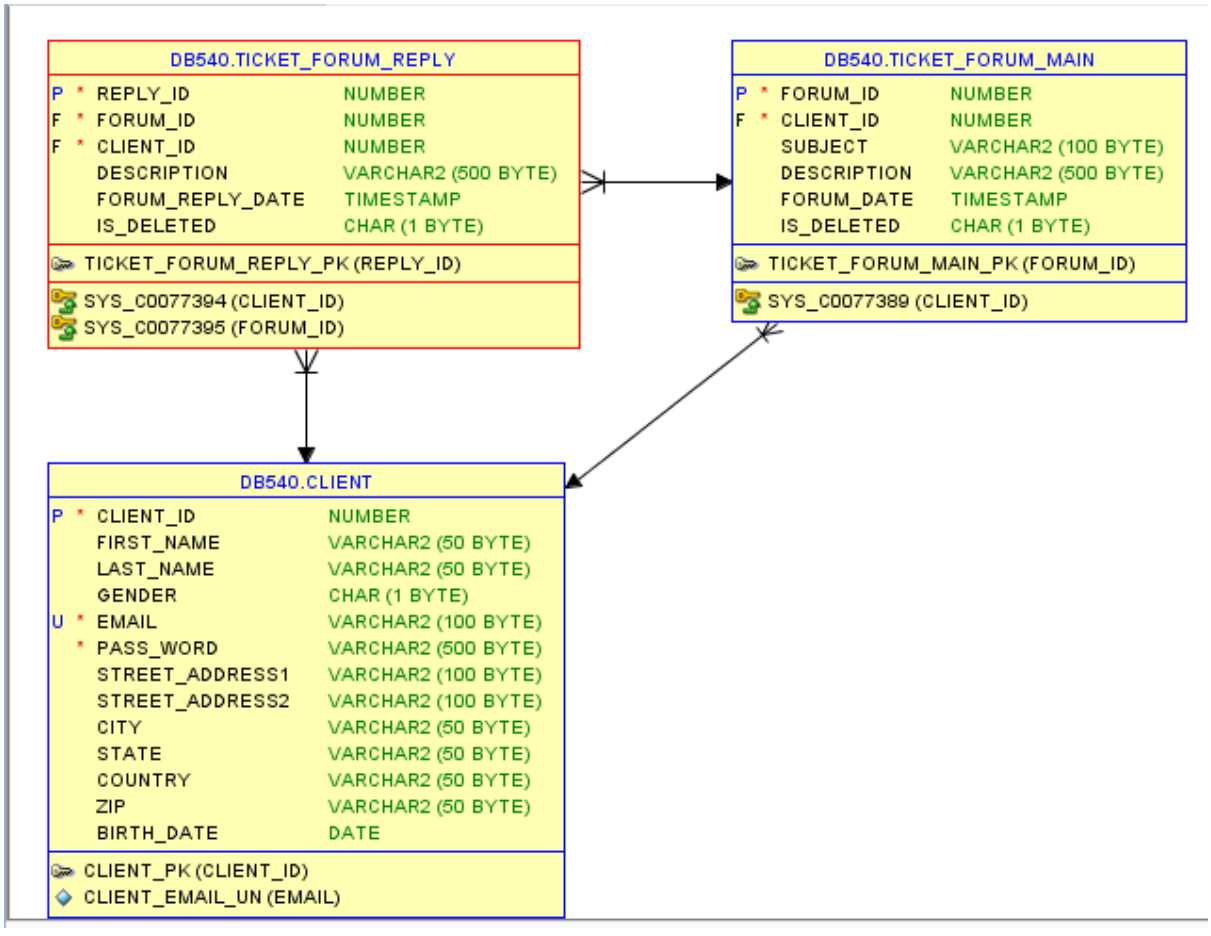


Figure 20 : Ticket Forum Reply - Model

Constraints

	CONSTRAINT_NAME	CONSTRAINT_TYPE	SEARCH_CONDITION	R_OWNER	R_TABLE_NAME	R_CONSTRAINT_NAME	DELETE_RULE	STATUS	DEFERRABLE	VALIDATED
1	SYS_C0077390	Check	"REPLY_ID" IS NOT NULL	(null)	(null)	(null)	(null)	ENABLED	NOT DEFERRABLE	VALIDATED
2	SYS_C0077391	Check	"FORUM_ID" IS NOT NULL	(null)	(null)	(null)	(null)	ENABLED	NOT DEFERRABLE	VALIDATED
3	SYS_C0077392	Check	"CLIENT_ID" IS NOT NULL	(null)	(null)	(null)	(null)	ENABLED	NOT DEFERRABLE	VALIDATED
4	SYS_C0077393	Primary_Key	(null)	(null)	(null)	(null)	(null)	ENABLED	NOT DEFERRABLE	VALIDATED
5	SYS_C0077394	Foreign_Key	(null)	DB540	CLIENT	SYS_C0077115	NO ACTION	ENABLED	NOT DEFERRABLE	VALIDATED
6	SYS_C0077395	Foreign_Key	(null)	DB540	TICKET_FORUM_MAIN	SYS_C0077388	NO ACTION	ENABLED	NOT DEFERRABLE	VALIDATED

Figure 21 : Ticket Forum Reply – Constraints

TABLE QUERIES

Client

Creation

```
CREATE TABLE CLIENT
(
  CLIENT_ID NUMBER NOT NULL PRIMARY KEY
, FIRST_NAME VARCHAR2(50 BYTE)
, LAST_NAME VARCHAR2(50 BYTE)
, GENDER CHAR(1 BYTE)
, EMAIL VARCHAR2(100 BYTE) NOT NULL
, PASS_WORD VARCHAR2(500 BYTE) NOT NULL
, STREET_ADDRESS1 VARCHAR2(100 BYTE)
, STREET_ADDRESS2 VARCHAR2(100 BYTE)
, CITY VARCHAR2(50 BYTE)
, STATE VARCHAR2(50 BYTE)
, COUNTRY VARCHAR2(50 BYTE)
, ZIP VARCHAR2(50 BYTE)
, BIRTH_DATE DATE
);
```

Sequence

```
CREATE SEQUENCE client_seq
  START WITH 1
  INCREMENT BY 1;
```

Constraints

```
ALTER TABLE Client
ADD UNIQUE (Email);
```

Staff

Creation

```
CREATE TABLE Staff
(
  Staff_ID NUMBER NOT NULL PRIMARY KEY
, User_Name VARCHAR2(100 BYTE)
, Pass_word VARCHAR2(500 BYTE)
, Is_Active CHAR(1 BYTE) );
```

Sequence

```
CREATE SEQUENCE staff_seq
  START WITH 1
  INCREMENT BY 1;
```

Ticket Master

Creation

```
CREATE TABLE Ticket_Master
(
  Ticket_ID NUMBER NOT NULL PRIMARY KEY
, Client_ID NUMBER NULL --FOREIGN KEY REFERENCES Client(Client_ID)
, Created_Date timestamp
, Ticket_Source varchar2(10 BYTE)
, Categories varchar2(50 BYTE)
, Assign_To Number NULL --FOREIGN KEY REFERENCES Staff(Staff_ID)
, Ticket_Status varchar2(50 BYTE)
, Priority char(1 BYTE)
, Closed_By Number
, Closed_Date timestamp
, Is_Deleted char(1 Byte) );
```

Sequence

```
CREATE SEQUENCE ticket_master_seq
```

```
  START WITH 1
  INCREMENT BY 1;
```

Constraints

```
ALTER TABLE TICKET_MASTER
ADD FOREIGN KEY (Client_ID) REFERENCES Client(Client_ID);

ALTER TABLE TICKET_MASTER
ADD FOREIGN KEY (Assign_To) REFERENCES Staff(Staff_ID);

ALTER TABLE TICKET_MASTER
MODIFY Priority number;
```

Ticket Activity

Creation

```
CREATE TABLE Ticket_Activity
(
  Activity_ID NUMBER NOT NULL PRIMARY KEY
, Ticket_ID NUMBER NULL -- FOREIGN KEY Ticket Master
, Activity_Type varchar2(50 BYTE)
, Subject varchar2(100 BYTE)
, Description varchar2(500 BYTE)
, Activity_Date timestamp
, Acted_By char(1 BYTE) -- Client C or Staff S
, Actor_ID number -- Client ID or Staff ID
, Unread char(1 Byte) );
```

Sequence

```
CREATE SEQUENCE ticket_activity_seq
  START WITH 1
```

INCREMENT BY 1;

Constraints

ALTER TABLE TICKET_ACTIVITY

ADD FOREIGN KEY (Ticket_ID) REFERENCES Ticket_Master(Ticket_ID);

ALTER TABLE TICKET_ACTIVITY

ADD FOREIGN KEY (ACTOR_ID) REFERENCES STAFF(STAFF_ID);

Blocklist

Creation

CREATE TABLE Blocklist

(

Client_ID NUMBER NOT NULL PRIMARY KEY, -- PRIMARY KEY FOREIGN KEY CLIENT ID

Staff_ID NUMBER NOT NULL, --FOREIGN KEY TICKET STAFF ID

DESCRIPTION VARCHAR2(500 BYTE) NULL

);

ALTER TABLE Blocklist

ADD Is_Deleted CHAR;

Sequence

CREATE SEQUENCE Blocklist_SEQ

START WITH 1

INCREMENT BY 1;

Constraints

ALTER TABLE Blocklist

ADD FOREIGN KEY (Client_ID) REFERENCES Client(Client_ID);

ALTER TABLE Blocklist

ADD FOREIGN KEY (STAFF_ID) REFERENCES STAFF(STAFF_ID);

Ticket Forum Main

Creation

```
CREATE TABLE Ticket_Forum_Main
(
  Forum_ID NUMBER NOT NULL PRIMARY KEY
, Client_ID NUMBER NOT NULL -- FOREIGN KEY Client
, Subject varchar2(100 BYTE)
, Description varchar2(500 BYTE)
, Forum_Date timestamp
, Is_Deleted char(1 Byte) );
```

Sequence

```
CREATE SEQUENCE ticket_forum_main_seq
  START WITH 1
  INCREMENT BY 1;
```

Constraints

```
ALTER TABLE Ticket_Forum_Main
ADD FOREIGN KEY (Client_ID) REFERENCES Client(Client_ID);
```

Ticket Forum Reply

Creation

```
CREATE TABLE Ticket_Forum_Reply
(
  Reply_ID NUMBER NOT NULL PRIMARY KEY
, Forum_ID NUMBER NOT NULL -- FOREIGN KEY Ticket_Forum_Main
, Client_ID NUMBER NOT NULL -- FOREIGN KEY Client
, Description varchar2(500 BYTE)
, Forum_Reply_Date timestamp
```

, Is_Deleted char(1 Byte));

Sequence

```
CREATE SEQUENCE ticket_forum_reply_seq
  START WITH 1
  INCREMENT BY 1;
```

Constraints

```
ALTER TABLE Ticket_Forum_Reply
ADD FOREIGN KEY (Client_ID) REFERENCES Client(Client_ID);

ALTER TABLE Ticket_Forum_Reply
ADD FOREIGN KEY (Forum_ID) REFERENCES Ticket_Forum_Main(Forum_ID);
```

DATA GENERATION AND LOADING

Sr.No.	Table Name	Query	Count(*)
1.	CLIENT	select count(*) from client;	11006
2.	STAFF	select count(*) from staff;	4
3.	BLOCKLIST	select count(*) from blocklist;	10168
4.	TICKET_MASTER	select count(*) from ticket_master;	4
5.	TICKET_ACTIVITY	select count(*) from ticket_activity;	14
6.	TICKET_FORUM_MAIN	select count(*) from ticket_forum_main;	2
7.	TICKET_FORUM_REPLY	select count(*) from ticket_forum_reply;	4

Table 9 : Data Generation 1

Inserted data manually in Staff, Ticket_Master, Ticket_Activity, Ticket_Forum_Main and Ticket_Forum_Reply. For performance tuning, we had to load more data in database, therefore the number of rows after loading data for performance tuning were:

Sr.No.	Table Name	Query	Count(*)
1.	CLIENT	select count(*) from client;	10909
2.	STAFF	select count(*) from staff;	888
3.	BLOCKLIST	select count(*) from blocklist;	10168
4.	TICKET_MASTER	select count(*) from ticket_master;	6000
5.	TICKET_ACTIVITY	select count(*) from ticket_activity;	60000
6.	TICKET_FORUM_MAIN	select count(*) from ticket_forum_main;	60000

7.	TICKET_FORUM_REPLY	select count(*) from ticket_forum_reply;	60000

Table 10 : Data Generation 2

We only increased number of rows to this extent to do performance tuning, therefore the rest of the details on the database are based on “Table 9 : Data Generation 1” and not on “Table 10 : Data Generation 2”.

Inserted data initially, manually in Client and Blocklist. Later on, inserted more than 10,000 rows using “mockaroo.com”. “Mockaroo” limits the number of rows to be generated to 1000 for free accounts, therefore had to change the ID every time and download the scripts for two tables more than 10 times each.

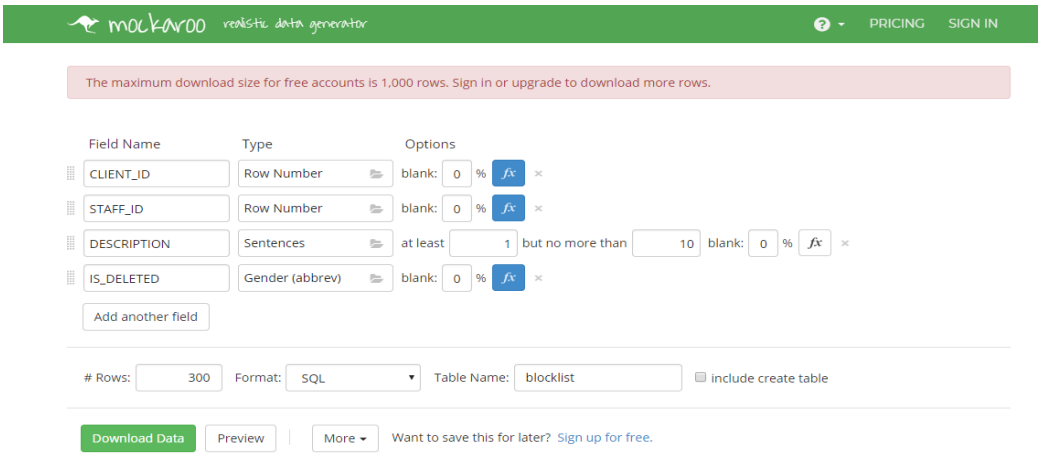


Figure 22 : Mockaroo Screenshot

The biggest advantage of Mackaroo is that you can write function to control the Field Value. For example: In gender,we required only ‘M’ for Male and ‘F’ for Female, but Mackaroo provided only “Male” and “Female” in gender. Tehrfore we wrote a function:

```
if this == 'Male' then
        'M’
else
        'F'
end
```

Also for primary key and foreign key constraints, they should be same while producing the data, therefore we can write a function for primary key:

```
this + 1024
```

But for loading data for performance tuning, we used a python script. As the data was large, it took 2 days to completely load the data in the database.

Also created a trigger to check Client's date of birth is later than Jan 1, 1900 and earlier than today.

```
CREATE OR REPLACE TRIGGER check_client_birth_date
  BEFORE INSERT OR UPDATE ON client
  FOR EACH ROW
BEGIN
  IF( :new.birth_date < date '1900-01-01' or
     :new.birth_date > sysdate )
  THEN
    RAISE_APPLICATION_ERROR(
      -20001,
      'Client date of birth must be later than Jan 1, 1900 and earlier than today' );
  END IF;
END;
```

PHYSICAL DATABASE DESIGN

The project designs a Ticket Management database which would be linked to a web page and provide information to web users. The client can register themselves and login through their user name and passwords. Clients are provided with 'Forgot Password', 'Change Password' and 'Update Profile' facilities. It is assumed that the database management staff will in turn use the information collected through the clients for analytics and extracting useful information from collected data.

There are broadly five different portals in this system:

1. Client:

- Can request for services via an interactive web portal.
- Post any questions or queries to the forum page provided.
- File a complaint against any staff member.

2. Staff:

- Communicate with the clients with acknowledgement and update ticket fulfillment progress including technician assignment
- The task assigned to the staff member is expected to be fulfilled in order to attain customer satisfaction.
- It also converts service requests via email into tickets.

3. Forum:

- Clients can post any questions or suggestions on this forum page.
- Clients can post as many replies as they want.

4. Ticket:

- The staff member can view all the activities related to a client's ticket id.
- A staff member can also a lot himself to a client.

5. Block list:

- A client can be blocked from Creating New Tickets
- We can view the list of customers that are blocked.

The physical design of the database depends on the frequency of the queries and the size of the database.

Predicted Usage (Growth Rate):

In the ticket management database, the important entity is tickets. It is inter-linked with many other entities in the database. Our database has 11006 clients in the TICKET_MASTER.CLIENT table. As the number of clients increase, the number of tickets generated will increase.

Considering the entire schema, the tables will automatically grow if a new client is added in the client table which in turn will include tables on 'STAFF', 'TICKET MASTER', 'TICKET ACTIVITY' , 'BLOCKLIST' as well as 'FORUM'.

One more interesting entity is the forum entity. Currently the database may not have a lot of posts on the forum page. But as the number of clients in the system increases, the number of posts they post and in turn the number of replies posted by the staff members increases. This shall entirely depend on how popular the web portal becomes and how much is the increase in its utility. If one entity of the database increases, it will definitely have a rapid increase in the other entities.

If we talk about the staff entity, it will remain comparatively constant until and unless a new employee joins the system. However, this will be updated almost every week to add a new staff member in case he joins the system.

Capacity Planning:

- Space occupied ,no of records, blocks:

The table below shows the storage structure of the ticket management database. The largest table is the client table which occupies 11006 KB and the smallest table is the Ticket_Forum_Main which occupies 2KB.

TABLE	SIZE(KB)
CLIENT	11006
STAFF	4
TICKET_ACTIVITY	14
TICKET_FORUM_MAIN	2
TICKET_FORUM_REPLY	4
TICKET_MASTER	4
BLOCKLIST	10168

Table 11 : Capacity

Table name	No. of rows	Blocks	Avg Row Length
CLIENT	11006	244	88
STAFF	4	5	35
TICKET_ACTIVITY	14	5	109
TICKET_FORUM_MAIN	2	5	81
TICKET_FORUM_REPLY	4	5	54
TICKET_MASTER	4	5	45
BLOCKLIST	10168	370	229

Table 12 : Capacity Detials

Query to be performed:

SELECT table_name, num_rows, blocks, avg_row_len from all_tables

WHERE owner='DB540';

According to the above data pulled from the tables, it is clear that the average row length is 91. bytes. So, assuming 500 more users use this system, the average row length would be 500*91 =45,500.

Thee growth and space occupied by the system can be tracked and monitored through the analysis done above.

Tables Views and Data files:

All the data is stored logically in table spaces as the database is utilizing oracle DBMS.

Data files will be associated with the tablespaces. . Every database has one primary data file. The primary data file has the startup information for the database and will refer to the other files in the database. All of the user data and objects can be stored in this file or in secondary data files

SQL Server files have the ability to grow automatically from their originally specified size. While defining a file, we can specific a growth increment. Every time the file is filled, it increases its size by the growth increment

Table spaces are one or more logical storage units in the database. Tablespaces are divided into logical units of storage called **segments**, which are further divided into **extents**. Extents are a collection of contiguous blocks.

The system table space is created by default in Oracle when a database is uploaded. From the point of view of current database this will facilitate in following activities:

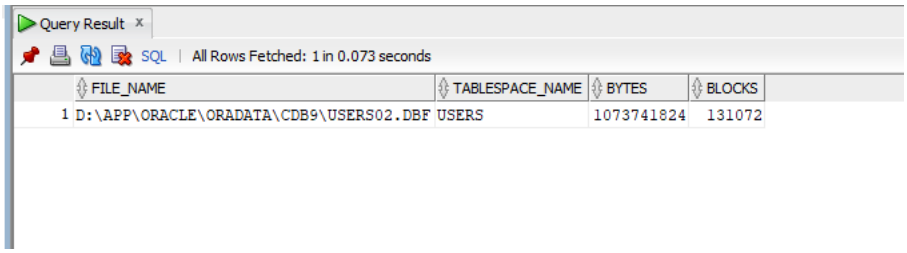
a) To assign a specific ticket to registered clients of the database.

b) User's privacy is also taken into consideration. They can update passwords and maintain security as and when they need.

c) Partial database backup and recovery operations can be executed.

```
SELECT
    file_name, tablespace_name,
    bytes,blocks
FROM dba_data_files
WHERE file_id= 9;
```

OUTPUT:



	FILE_NAME	TABLESPACE_NAME	BYTES	BLOCKS
1	D:\APP\ORACLE\ORADATA\CDB9\USERS02.DBF	USERS	1073741824	131072

Figure 23 : dba_data_files

The query written below shows more details about the table spaces:

```
SELECT TABLESPACE_NAME"TABLESPACE",FILE_ID,
  COUNT(*) "PIECES",
  MAX(blocks)"MAXIMUM",
  MIN(blocks)"MINIMUM",
  AVG(blocks)"AVERAGE",
  SUM(blocks)"TOTAL"
FROM DBA_FREE_SPACE
GROUPBY TABLESPACE_NAME, FILE_ID;
```

OUTPUT:

	TABLESPACE	FILE_ID	PIECES	MAXIMUM	MINIMUM	AVERAGE	TOTAL
1	UNDOTBS1	5	10	363264	16	36730.4	367304
2	STUDENTS	16	23	1920	8	216	4968
3	EXAMPLE	7	2	5200	128	2664	5328
4	STUDENTS	20	12	7168	128	1696	20352
5	STUDENTS	14	75	1024	8	94.72	7104
6	STUDENTS	18	32	896	8	82	2624
7	STUDENTS	17	68	1280	8	77.764705882352941176470588235294117647058	5288
8	COLORS	15	56	35552	8	724.142857142857142857142857142857142857	40552
9	COLORS	12	77	77824	8	1071.064935064935064935064935064935064935	82472
10	USERS	9	59	1024	8	75.932203389830508474576271186440677966	4480
11	SYSAUX	3	1353	5056	8	13.835920177383592017738359201773835920	18720
12	USERS	23	3	84224	80	28144	84432
13	STUDENTS	10	94	768	8	23.148936170212765957446808510638297872340	2176
14	STUDENTS	21	38	104	8	27.368421052631578947368421052631578947370	1040
15	COLORS	11	73	38656	8	578.739726027397260273972602739726027397	42248
16	COLORS	22	1	130944	130944	130944	130944
17	SYSTEM	1	3	1152	8	394.66666666666666666666666666666666666667	1184
18	USERS	24	1	77824	77824	77824	77824
19	STUDENTS	19	48	80	8	16.833333333333333333333333333333333333	808
20	USERS	6	431	1024	8	28.194895591647331786542923433847099768	12152
21	STUDENTS	13	82	1280	8	91.219512195121951219512195121951219512195122	7480
22	STUDENTS	8	170	512	8	22.258823529411764705882352941176470588235294	3784

Figure 24 : DBA_FREE_SPACE

From the above output, the ‘Pieces’ shows the number of free spaces extents in the table space. ‘Maximum’ and ‘Minimum’ shows the largest and the smallest contiguous area of space in data blocks. ‘Average’ shows the average size in blocks of a free space extent. The Total gives the total of free space in each of the shown table space files in blocks. The information extracted from this query can be utilized when there shall be more entries in the table space.

Partitioning:

The process of decomposing a large table into small and manageable segments is called Partitioning. It helps in query processing. Our ticket management database is not so large i.e the tables are not large enough to be partitioned. But, there are chances that the number of clients and henceforth the number of tickets gets raised, in that scenario we would be requiring partitioning to be done. In that case here are some of the uses of implementing partitioning in our database:

- It will facilitate query processing when dealing with large tables.
- It will provide us with data recovery and backup.
- It will help us with maintenance on one partition without affecting other partitions.
- Maintenance operations can be implemented quickly.

PERFORMANCE TUNING

Below are a few use cases for our database which stores tickets created for a customer support centre.

User:

- To get information about a client's previous/existing tickets.
- To find all forum posts that a client has made.
- To look for all the tickets that are assigned to them.

Manager:

- To check all active/pending tickets.
- To find all tickets that were created within the past few days

As the tickets table is the most used table in the database with the tickets being the main source of information about a client's issue and them being regularly retrieved we will start indexing the tickets table and see how that impacts the performance of the database.

B-Tree Indexing: B-Tree Indexing is a balanced tree indexing with a tree like index structure. It is good for high cardinality attributes and when there are many point and range queries. Some of the disadvantages of using a B+ Tree index are that they require some overhead when inserting and deleting and they also require some space overhead.

Bitmap Indexing: Bitmap indexing is a type of indexing that is commonly used in data warehousing. It is efficient when querying on multiple keys and is great for tables in which changes rarely occur. It comprises of an array of bits.

For our table we are aware that our ticket, ticket activity, ticket forum, ticket forum reply tables are all regularly updated with new values whereas the client table and staff table need to be updated very rarely. Based on this knowledge about our database system we would be utilizing B+ Index on the tables that are regularly updated, and we would be utilizing bitmap index on the tables that need not be regularly updated.

Now let us look at some of the most used queries and their performance stats before and after completing indexing.

Example 1: Show all Tickets to a Staff member

```
SELECT
   TICKET_MASTER.TICKET_ID,
   TICKET_MASTER.CLIENT_ID,
   TICKET_MASTER.ASSIGN_TO,
   TICKET_MASTER.CREATED_DATE,
   TICKET_MASTER.TICKET_STATUS
```

```
FROM
   TICKET_MASTER
WHERE
   TICKET_MASTER.ASSIGN_TO = '647'
ORDER BY
   TICKET_MASTER.PRIORITY,
   TICKET_MASTER.CREATED_DATE;
```

Query Execution Plan:

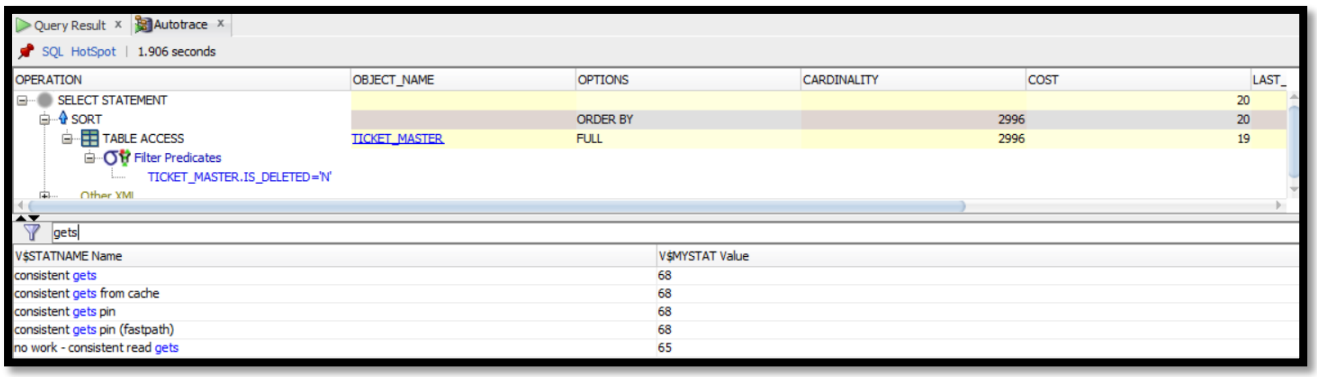


Figure 25 : Show all Tickets to a Staff member

Query Execution Plan after Indexing:

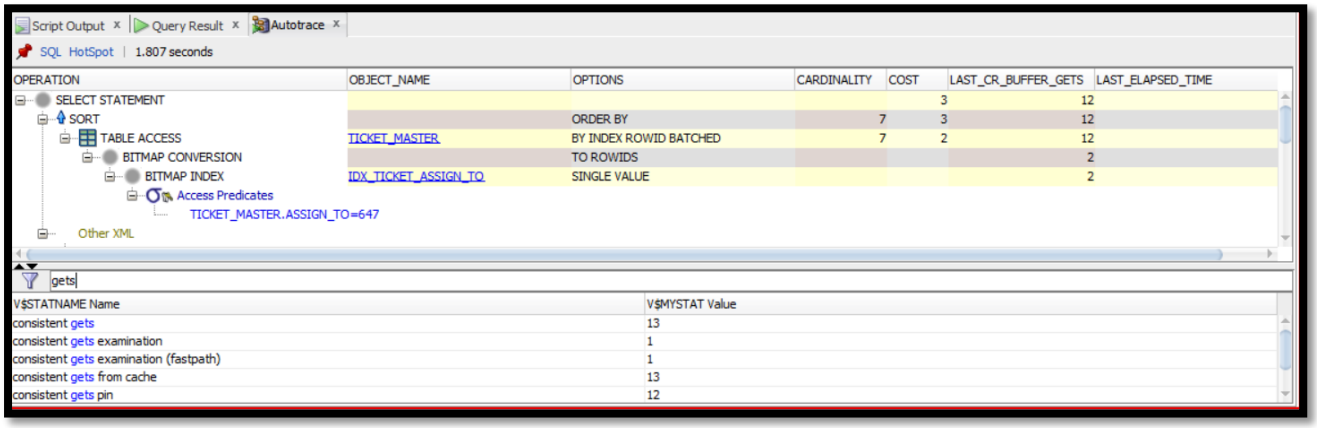


Figure 26 : Show all Tickets to a Staff member with Indexing

Example 2: Display all activities performed on a specific ticket

Query:

```
   SELECT
   TICKET_ACTIVITY.ACTIVITY_TYPE,
   TICKET_ACTIVITY.ACTIVITY_DATE,
   TICKET_ACTIVITY.ACTED_BY,
   TICKET_ACTIVITY.DESCRIPTION
FROM
   TICKET_ACTIVITY
WHERE
```

TICKET_ACTIVITY.TICKET_ID = 4725
ORDER BY
TICKET_ACTIVITY.ACTIVITY_DATE

Query Execution Plan:

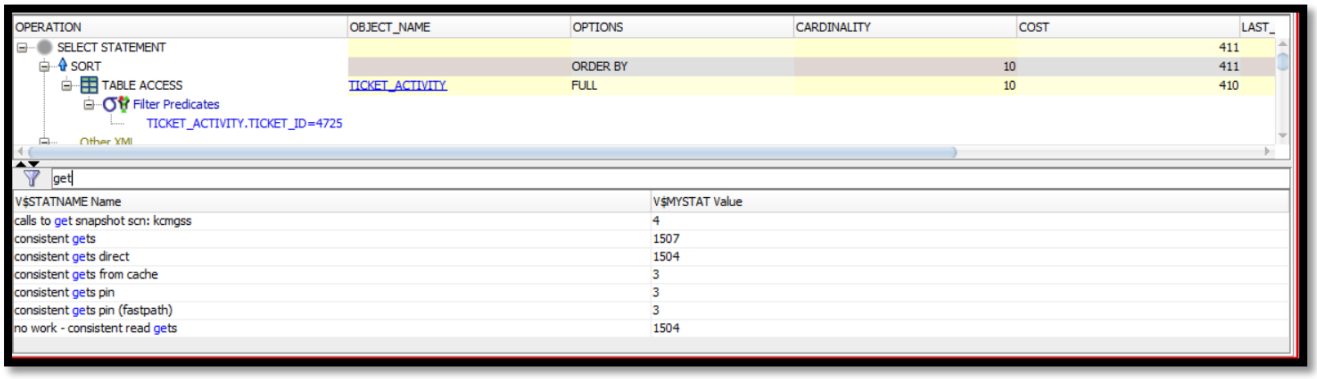


Figure 27 : Display all activities performed on a specific ticket

Query Execution Plan after Indexing:

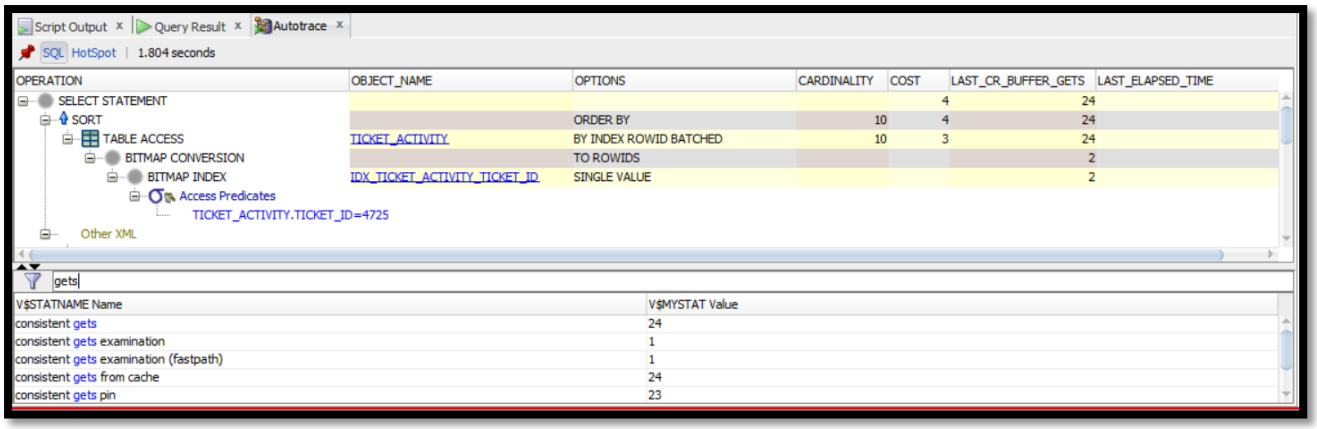


Figure 28 : Display all activities performed on a specific ticket with Indexing

Example 3: Need to fetch the records for all tickets that have been created in the past 2 days.

```
select * from ticket_master
where CREATED_DATE >= sysdate -2
order by CREATED_DATE desc;
```

Query Execution Plan:

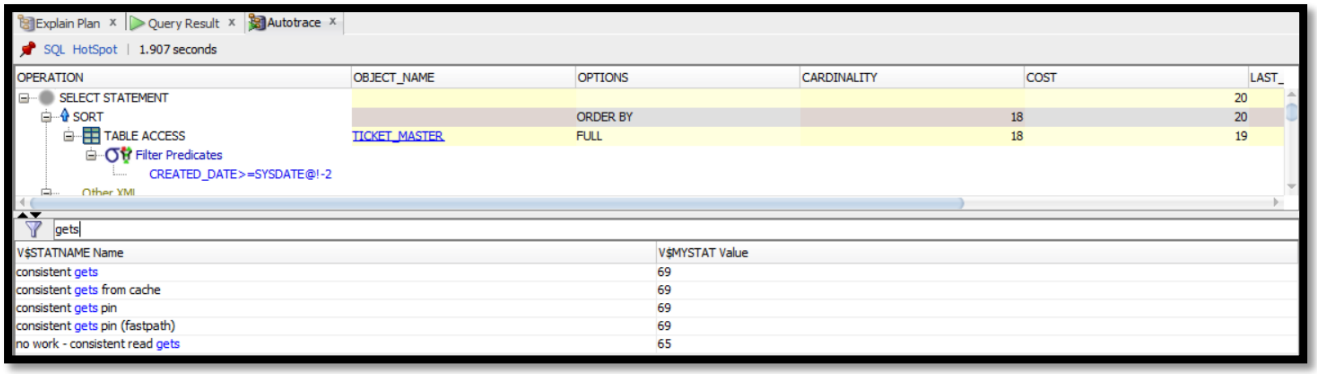


Figure 29 : all tickets created in past 2 days

Query Execution Plan after Indexing:

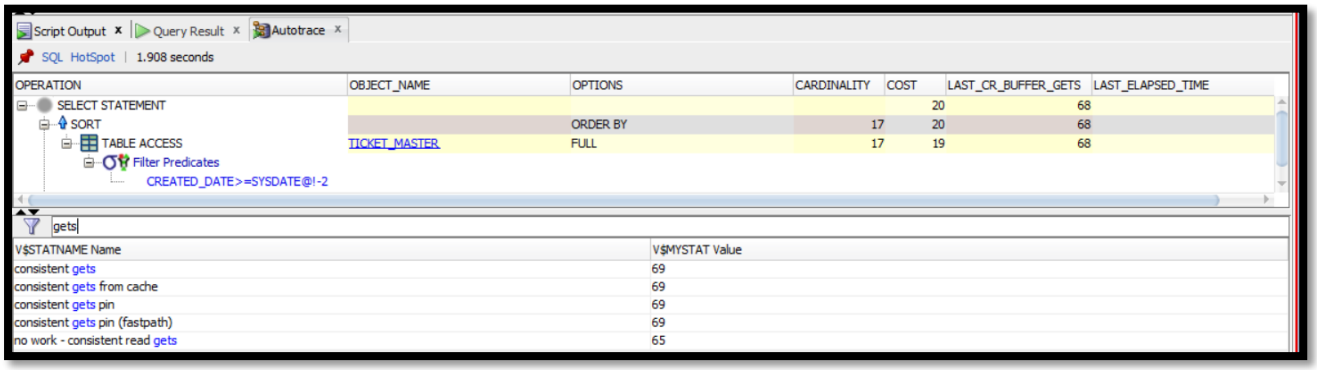


Figure 30 : all tickets created in past 2 days with Indexing

Example 4: Fetch records in the last 2 days for tickets that are in closed state.

Query:

```
select * from ticket_master
where CREATED_DATE >= sysdate -2 and ticket_status like '%Close%'
order by CREATED_DATE desc;
```

Query Execution Plan:

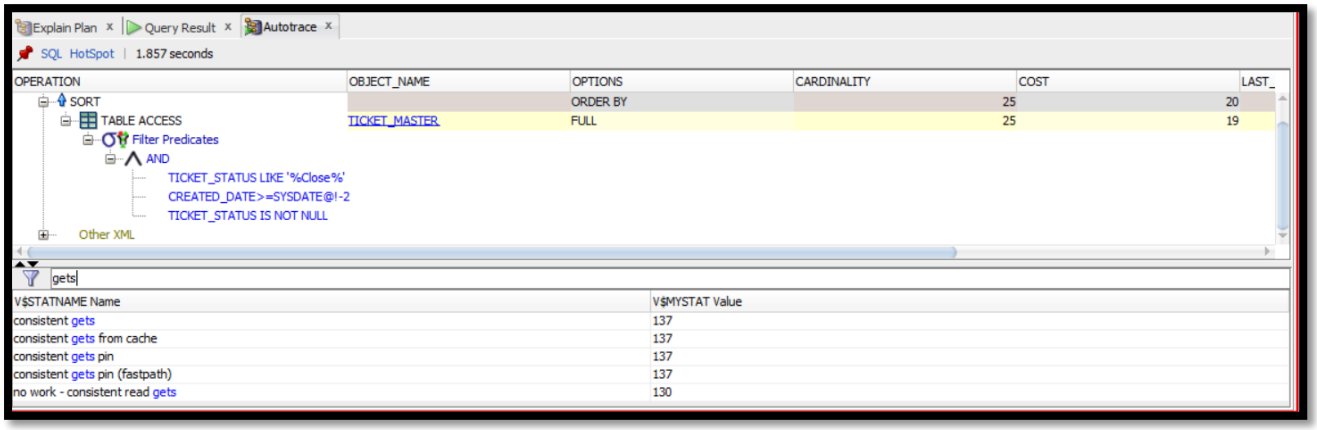


Figure 31 : tickets created in past 2 day with 'Closed' Status

Query Execution Plan after Indexing:

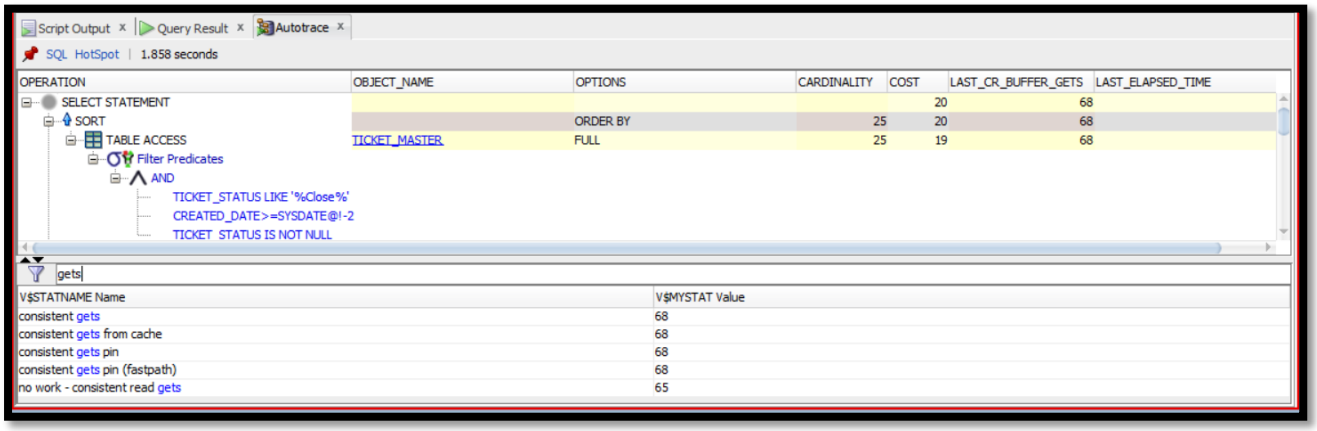


Figure 32 : tickets created in past 2 day with 'Closed' Status with Indexing

Example 5: Fetch records for all customers whose first name is “Sean”

Query:

```
SELECT
   TICKET_MASTER.TICKET_ID,
   TICKET_ACTIVITY.ACTIVITY_ID,
   CLIENT.First_name,
   CLIENT.EMAIL,
   --TICKET_MASTER.CLIENT_ID,
   TICKET_ACTIVITY.SUBJECT,
   --STAFF.USER_NAME,
   TICKET_ACTIVITY.ACTIVITY_TYPE,
   TICKET_ACTIVITY.ACTOR_ID,
   TICKET_ACTIVITY.DESCRIPTION,
   TICKET_ACTIVITY.ACTIVITY_DATE,
   TICKET_MASTER.TICKET_STATUS
FROM
    TICKET_ACTIVITY
```

```
    LEFT OUTER JOIN  TICKET_MASTER
       ON TICKET_ACTIVITY.TICKET_ID = TICKET_MASTER.TICKET_ID
    LEFT OUTER JOIN CLIENT
       ON TICKET_MASTER.CLIENT_ID = CLIENT.CLIENT_ID
WHERE
   TICKET_MASTER.IS_DELETED = 'N' AND CLIENT.FIRST_NAME like '%Sean%'
ORDER BY
   TICKET_ACTIVITY.ACTIVITY_DATE;
```

Query Execution Plan:

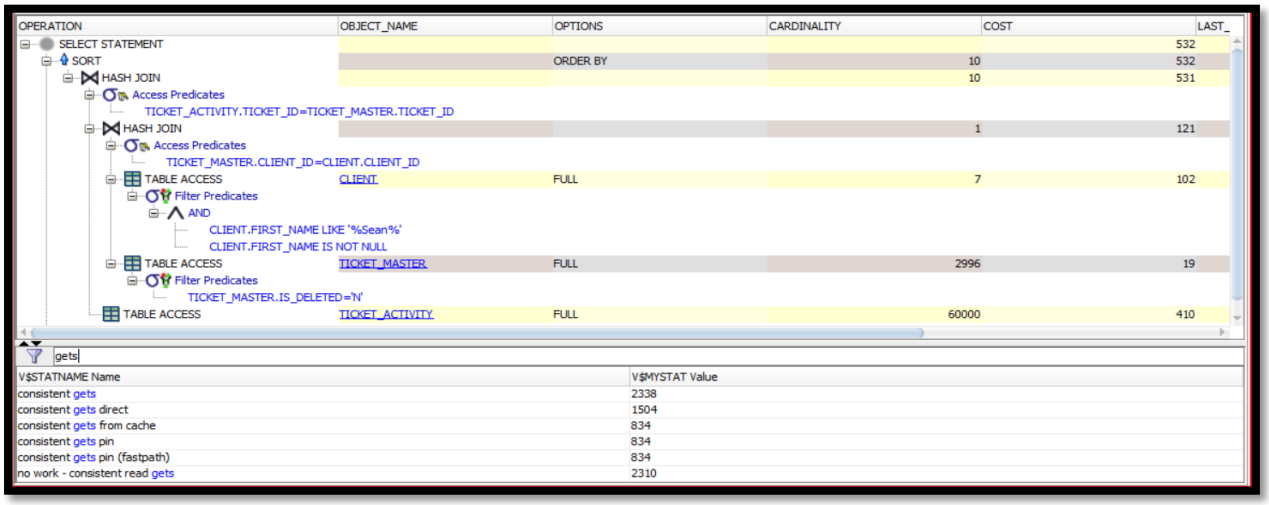


Figure 33 : all records of customers whose first name is "Sean"

Query Execution Plan after Indexing:

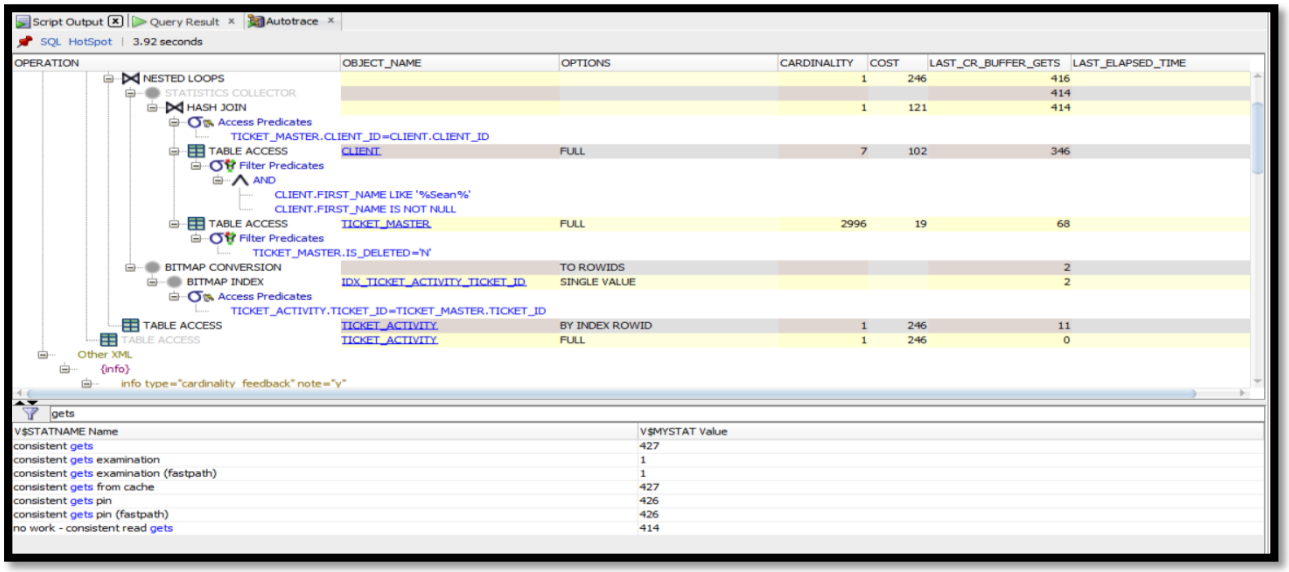


Figure 34 : all records of customers whose first name is "Sean" with Indexing

Example 5: Fetch records for all customers whose first name starts with “Ca”

Query:

```
SELECT
   TICKET_MASTER.TICKET_ID,
   TICKET_ACTIVITY.ACTIVITY_ID,
   CLIENT.First_name,
   CLIENT.EMAIL,
   --TICKET_MASTER.CLIENT_ID,
   TICKET_ACTIVITY.SUBJECT,
   --STAFF.USER_NAME,
   TICKET_ACTIVITY.ACTIVITY_TYPE,
   TICKET_ACTIVITY.ACTOR_ID,
   TICKET_ACTIVITY.DESCRIPTION,
   TICKET_ACTIVITY.ACTIVITY_DATE,
   TICKET_MASTER.TICKET_STATUS
FROM
    TICKET_ACTIVITY
      LEFT OUTER JOIN  TICKET_MASTER
         ON TICKET_ACTIVITY.TICKET_ID = TICKET_MASTER.TICKET_ID
      LEFT OUTER JOIN CLIENT
         ON TICKET_MASTER.CLIENT_ID = CLIENT.CLIENT_ID
WHERE
   TICKET_MASTER.IS_DELETED = 'N' AND CLIENT.FIRST_NAME like 'Ca%'
ORDER BY
   TICKET_ACTIVITY.ACTIVITY_DATE;
```

Query Execution Plan:

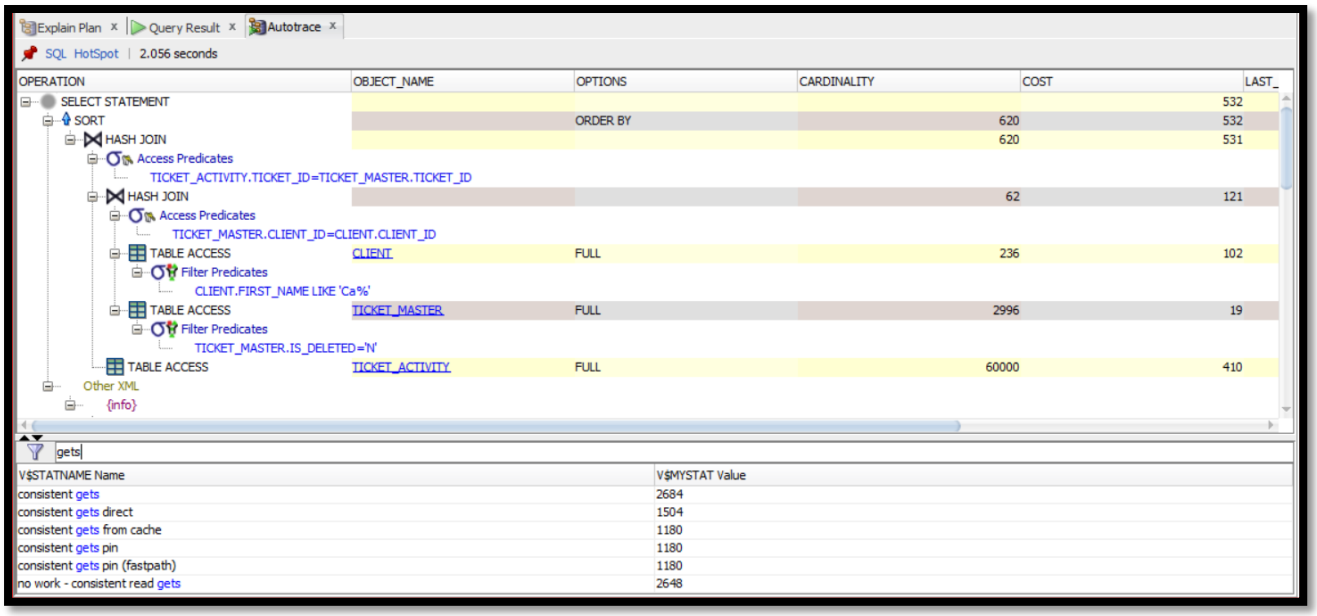


Figure 35 : all records of customers whose first name start with “Ca”

Query Execution plan after Indexing:

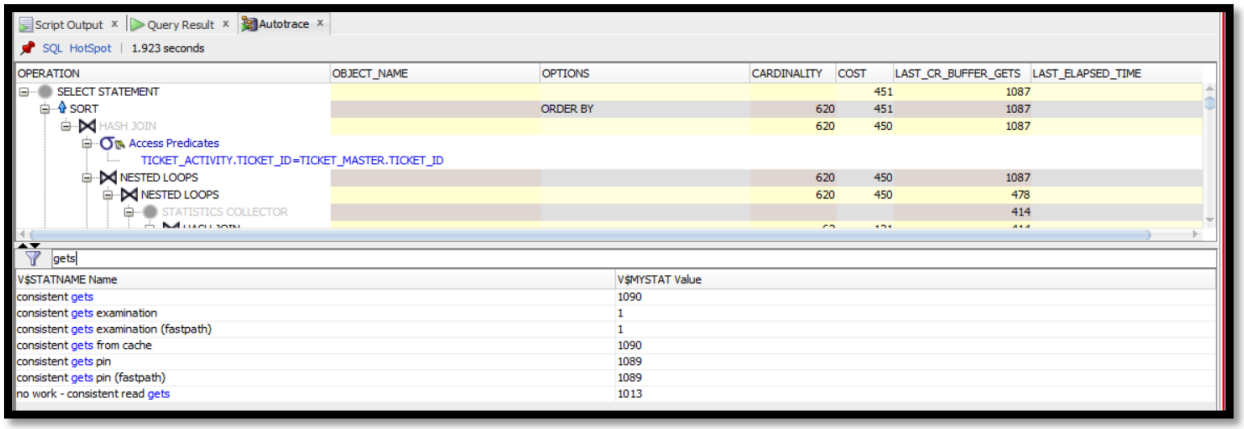


Figure 36 : all records of customers whose first name start with "Ca" with indexing

Example 6: Fetch all tickets that are in active ticket state

Query:

```
SELECT
   TICKET_MASTER.CLIENT_ID,
   TICKET_MASTER.TICKET_ID,
   --TICKET_ACTIVITY.ACTIVITY_ID,
   --TICKET_MASTER.CLIENT_ID,
   --TICKET_ACTIVITY.SUBJECT,
   --STAFF.USER_NAME,
   --TICKET_ACTIVITY.ACTIVITY_TYPE,
   TICKET_MASTER.TICKET_STATUS
FROM
    TICKET_MASTER
WHERE
   TICKET_MASTER.IS_DELETED = 'N' AND TICKET_MASTER.TICKET_STATUS
like '%Active%'
ORDER BY
   TICKET_MASTER.CREATED_DATE;
```

Query Execution Plan:

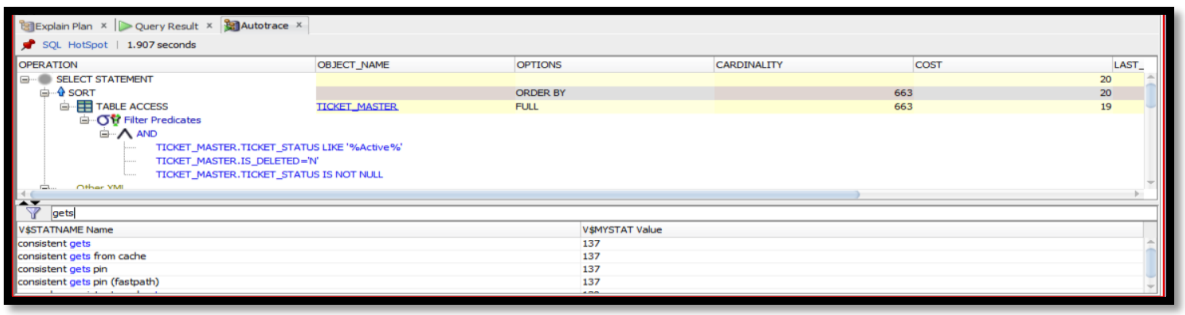


Figure 37 : all tickets with active status

Query Execution Plan after Indexing:

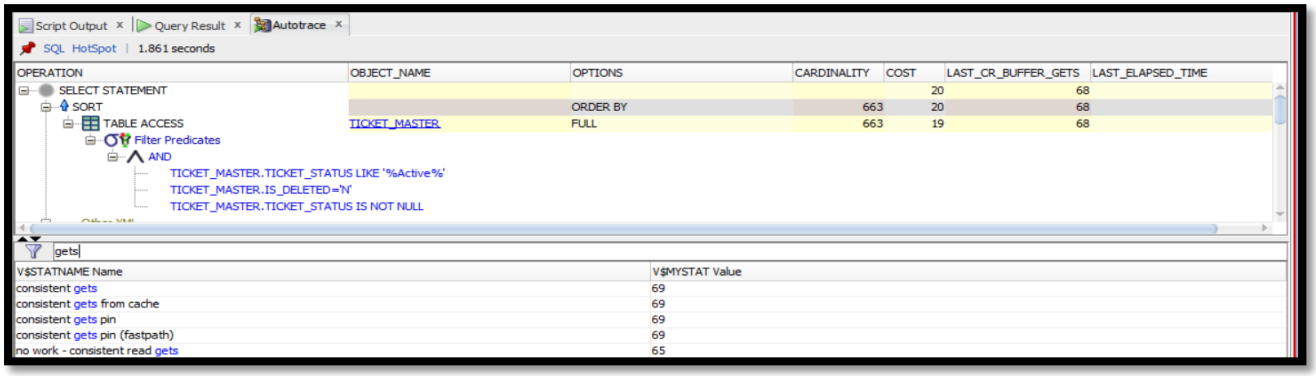


Figure 38 : all tickets with active status with Indexing

Example 7: Fetch all forum records for a user whose name is “Sean”

Query:

```
SELECT
   TICKET_Forum_main.forum_ID,
   TICKET_Forum_Reply.Reply_ID,
   CLIENT.First_name,
   TICKET_Forum_main.SUBJECT,
   TICKET_Forum_Reply.DESCRIPTION
FROM
    TICKET_Forum_Reply
      LEFT OUTER JOIN  TICKET_Forum_main
         ON TICKET_Forum_Reply.Forum_ID = TICKET_Forum_main.Forum_ID
         LEFT OUTER JOIN CLIENT
            ON TICKET_Forum_main.CLIENT_ID = CLIENT.CLIENT_ID
      LEFT OUTER JOIN CLIENT
         ON TICKET_Forum_Reply.CLIENT_ID = CLIENT.CLIENT_ID
WHERE
   CLIENT.FIRST_NAME like '%Sean%';
```

Query Execution Plan:

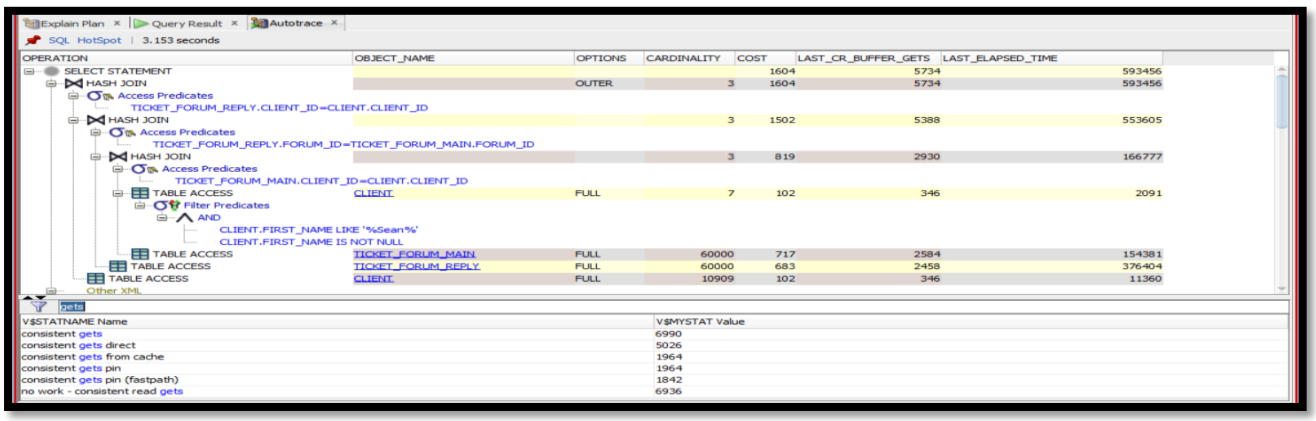


Figure 39 : all forum of customers whose name is 'Sean'

Query Execution Plan after Indexing:

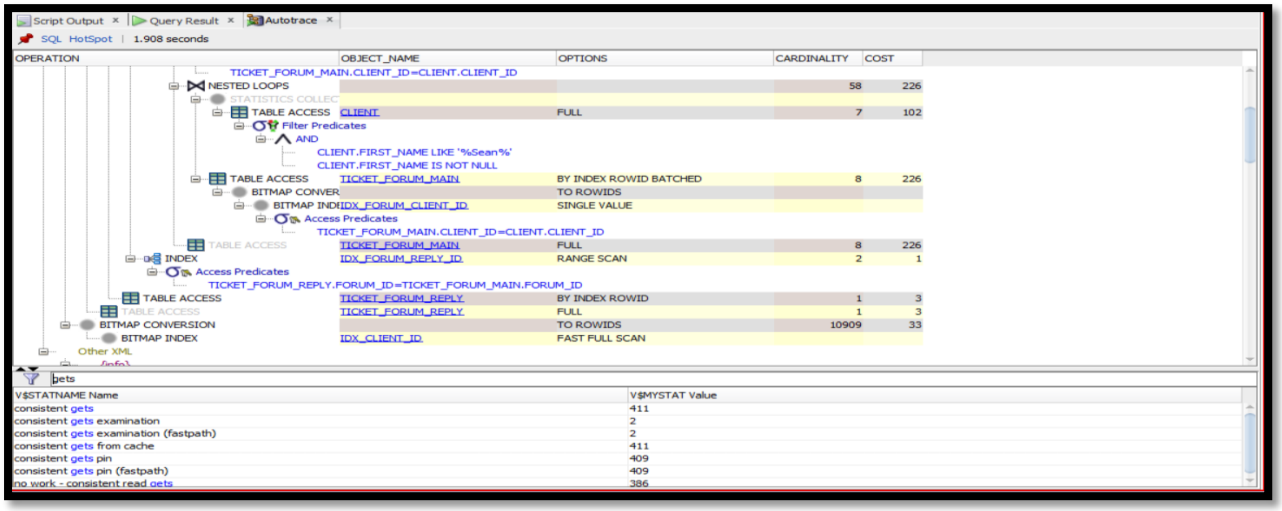


Figure 40 : all forum of customers whose name is 'Sean' with Indexing

Example 8: All tickets that are in active state for a particular user (Employee/Staff)

Query:

```
SELECT
   Staff.user_name,
   TICKET_MASTER.CLIENT_ID,
   TICKET_MASTER.TICKET_ID,
   TICKET_MASTER.TICKET_STATUS
FROM
    TICKET_MASTER
    inner join staff
      on TICKET_MASTER.ASSIGN_TO = staff.STAFF_ID
WHERE
   TICKET_MASTER.IS_DELETED = 'N' AND TICKET_MASTER.TICKET_STATUS
like '%Active%' AND staff.user_name like '%Larsen2023%';
```

Query Execution Plan:

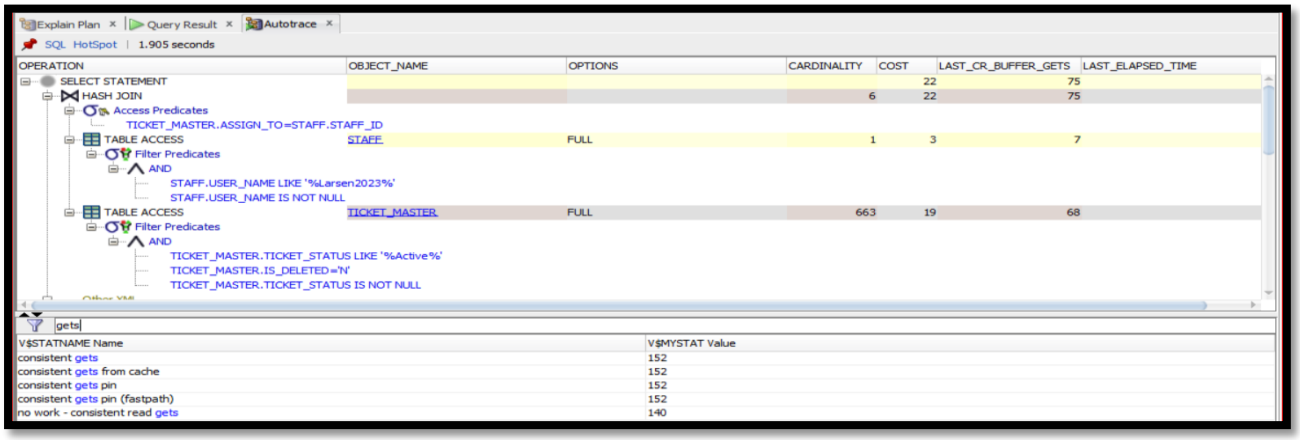


Figure 41 : all records where ticket is in active state for a staff

Query Execution plan after Indexing:

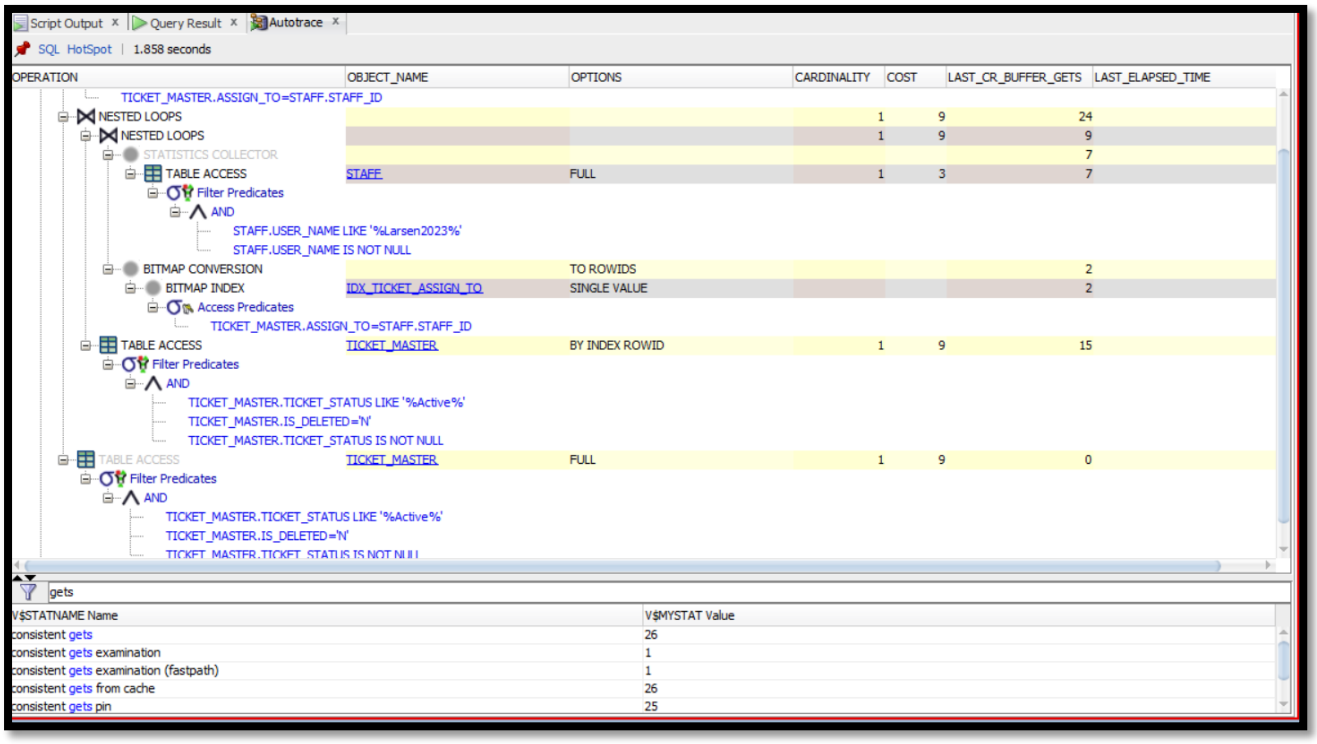


Figure 42 : all records where ticket is in active state for a staff with Indexing

Based on the above examples we can see that indexing has improved the performance of the database considerably. In order to further improve the efficiency of the database we can explore options of global partitioned indexes and partitioned tables along with bitmap join indexes.

DBA SCRIPTING:

The following are some of the scripts that can be useful for a DBA to monitor and manage database effectively:

1. Scripts to obtain information on ROWID's, FILE , OBJECTS or tablespace in which a table is stored.

```
SELECT
  ROWID, DBMS_ROWID.ROWID_OBJECT(ROWID)"OBJECT",
  DBMS_ROWID.ROWID_RELATIVE_FNO(ROWID)"FILE",
 DBMS_ROWID.ROWID_BLOCK_NUMBER(ROWID)"BLOCK",
  DBMS_ROWID.ROWID_ROW_NUMBER(ROWID)"ROW",
  client_id
FROM
  CLIENT;
```

OUTPUT:

	ROWID	OBJECT	FILE	BLOCK	ROW	CLIENT_ID
1	AABKUXAAVAAAGFNAAh	304407	21	24909	33	1613
2	AABKUXAAVAAAGFNAAi	304407	21	24909	34	1614
3	AABKUXAAVAAAGFNAAj	304407	21	24909	35	1615
4	AABKUXAAVAAAGFNAAk	304407	21	24909	36	1616
5	AABKUXAAVAAAGFNAAl	304407	21	24909	37	1617

Figure 43 : CLIENT

2. Script to fetch information on the ownership of an object:

```
SELECT
  owner,object_name, object_type,created
FROM
   dba_objects
WHERE
   object_id=304407;
```

OUTPUT:

	OWNER	OBJECT_NAME	OBJECT_TYPE	CREATED
1	DB540	CLIENT	TABLE	25-MAR-18

Figure 44 : dba_objects

3. Scripts for information on Tablespace:

```
SELECT
   file_name,tablespace_name, bytes,blocks
FROM
   dba_data_files
WHERE
  file_id=21;
```

OUTPUT:

	FILE_NAME	TABLESPACE_NAME	BYTES	BLOCKS
1	D:\APP\ORACLE\ORADATA\CDB9\STUDENTS10.DBF	STUDENTS	1073741824	131072

Figure 45 : dba_data_files

4. Scripts for getting information on all tables and space occupied by them:

```
SELECT
   table_name,num_rows,empty_blocks,avg_space,chain_cnt,avg_row_len
FROM
   all_tables
WHERE
  owner='DB540';
```

	TABLE_NAME	NUM_ROWS	EMPTY_BLOCKS	AVG_SPACE	CHAIN_CNT	AVG_ROW_LEN
1	BLOCKLIST	10168	0	0	0	229
2	CLIENT	11006	0	0	0	88
3	STAFF	4	0	0	0	35
4	TICKET_ACTIVITY	14	0	0	0	109
5	TICKET_FORUM_MAIN	2	0	0	0	81
6	TICKET_FORUM_REPLY	4	0	0	0	54
7	TICKET_MASTER	4	0	0	0	45

Figure 46 : all_tables

5. Script for analysing space available in various tablespaces in the database and maximum and minimum block size.

```
SELECT

  TABLESPACE_NAME"TABLESPACE",FILE_ID, COUNT(*) "PIECES", MAX(blocks)"MAXIMUM", MIN(blocks)"MINIMUM", AVG(blocks)"AVERAGE", SUM(blocks)"TOTAL"

FROM

  DBA_FREE_SPACE

GROUP BY

  TABLESPACE_NAME,FILE_ID;
```

	TABLESPACE	FILE_ID	PIECES	MAXIMUM	MINIMUM	AVERAGE	TOTAL
1	UNDOTBS1	5	9	363264	16	40811.555555555555555555555555555555556	367304
2	STUDENTS	16	22	1920	8	225.09090909090909090909090909090909	4952
3	EXAMPLE	7	2	5200	128	2664	5328
4	STUDENTS	20	12	7168	128	1685.3333333333333333333333333333333333	20224
5	STUDENTS	18	32	896	8	77.5	2480
6	STUDENTS	17	66	1280	8	77.818181818181818181818181818181818182	5136

Figure 47 : DBA_FREE

The following are some scripts related to indexes:

6. Script to list all the indexes of and database owner.

```
SELECT *

FROM

    dba_indexes

WHERE

    owner='DB540'

ORDER BY

  index_name;
```

OUTPUT:

	OWNER	INDEX_NAME	INDEX_TYPE	TABLE_OWNER	TABLE_NAME	TABLE_TYPE	UNIQUENESS	COMPRESSION	PREFIX_LENGTH	TABLESPACE_NAME	INI_TRANS	MAX_TRANS
1	DB540	SYS_C0077115	NORMAL	DB540	CLIENT	TABLE	UNIQUE	DISABLED	(null)	STUDENTS	2	255
2	DB540	SYS_C0077117	NORMAL	DB540	STAFF	TABLE	UNIQUE	DISABLED	(null)	STUDENTS	2	255
3	DB540	SYS_C0077123	NORMAL	DB540	TICKET_MASTER	TABLE	UNIQUE	DISABLED	(null)	STUDENTS	2	255
4	DB540	SYS_C0077127	NORMAL	DB540	TICKET_ACTIVITY	TABLE	UNIQUE	DISABLED	(null)	STUDENTS	2	255
5	DB540	SYS_C0077378	NORMAL	DB540	BLOCKLIST	TABLE	UNIQUE	DISABLED	(null)	STUDENTS	2	255
6	DB540	SYS_C0077388	NORMAL	DB540	TICKET_FORUM_MAIN	TABLE	UNIQUE	DISABLED	(null)	STUDENTS	2	255
7	DB540	SYS_C0077393	NORMAL	DB540	TICKET_FORUM_REPLY	TABLE	UNIQUE	DISABLED	(null)	STUDENTS	2	255
8	DB540	SYS_C0078200	NORMAL	DB540	CLIENT	TABLE	UNIQUE	DISABLED	(null)	STUDENTS	2	255

Figure 48 : dba_indexes

7. DBA scripts for finding constraints in the tables

```
SELECT*
FROM
   dba_cons_columns dba_constraints
WHERE
   owner='DB540';
```

OUTPUT:

	OWNER	CONSTRAINT_NAME	TABLE_NAME	COLUMN_NAME	POSITION
1	DB540	SYS_C0078200	CLIENT	EMAIL	1
2	DB540	SYS_C0077395	TICKET_FORUM_REPLY	FORUM_ID	1
3	DB540	SYS_C0077394	TICKET_FORUM_REPLY	CLIENT_ID	1
4	DB540	SYS_C0077393	TICKET_FORUM_REPLY	REPLY_ID	1
5	DB540	SYS_C0077392	TICKET_FORUM_REPLY	CLIENT_ID	(null)
6	DB540	SYS_C0077391	TICKET_FORUM_REPLY	FORUM_ID	(null)
7	DB540	SYS_C0077390	TICKET_FORUM_REPLY	REPLY_ID	(null)
8	DB540	SYS_C0077389	TICKET_FORUM_MAIN	CLIENT_ID	1
9	DB540	SYS_C0077388	TICKET_FORUM_MAIN	FORUM_ID	1
10	DB540	SYS_C0077387	TICKET_FORUM_MAIN	CLIENT_ID	(null)
11	DB540	SYS_C0077386	TICKET_FORUM_MAIN	FORUM_ID	(null)
12	DB540	SYS_C0077381	TICKET_ACTIVITY	ACTOR_ID	1

Figure 49 : dba_cons_columns

DATABASE SECURITY

There is a Password field in Client and Staff Table. Therefore, instead of storing password in text form, we store it in hash form. In case the database gets leaked the password will remain safe as there is tendency of using nearly same password by clients.

```
CREATE OR REPLACE FUNCTION "SHA" (p_string IN VARCHAR2, p_bits IN
NUMBER)
  RETURN RAW
AS
  LANGUAGE JAVA
  NAME 'sha.get_digest( java.lang.String, int ) return oracle.sql.RAW' ;

CREATE OR REPLACE AND COMPILE  JAVA SOURCE NAMED sha
  AS import java.security.MessageDigest;
import oracle.sql.*;
public class sha
{
public static oracle.sql.RAW get_digest( String p_string, int p_bits ) throws Exception
{
MessageDigest v_md = MessageDigest.getInstance( "SHA-" + p_bits );
byte[] v_digest;
v_digest = v_md.digest( p_string.getBytes( "UTF-8" ) );
return RAW.newRAW(v_digest);
}
}
/
```

Output:

```
--In 256 Encryption will get 64 Character Length String
SQL> SELECT LOWER(SHA('Oracle PL/SQL World',256)) FROM DUAL;
1f2f1fd5ab78cf15bd3cdf330d6ab79ba60cc797fb4875e5267f90daa10157aa
```

--In 512 Encryption will get 128 Character Length String

```
SQL> SELECT LOWER(SHA('Oracle PL/SQL World',512)) FROM DUAL;
478811ecf93ae1564ad0775c20aa0f1b5de5992306974a9339b25794de32a000d058136c48122
9c38f3ee9e6775426a3b54ff1daf495d3b7f2b6e571595ca888
```

QUERYING

Client Side

Client Registration

INSERT INTO CLIENT values

(CLIENT_SEQ.NEXTVAL,

'TONY',

'STARK',

'M',

'stark@stark.com',

(SELECT LOWER(SHA('stark@tony',256)) FROM DUAL),

",",",",",",",

DATE '1996-03-26');

Figure 50 : Client Registration

Client Login

```
SELECT
EMAIL, PASS_WORD
FROM CLIENT
WHERE
EMAIL LIKE 'mayanky@mail.usf.edu' AND
PASS_WORD LIKE (SELECT LOWER(SHA('stark@tony',256)) FROM DUAL) ;
```

Figure 51 : Client Login

Forgot Password

```
SELECT * FROM DUAL WHERE (SELECT LOWER(SHA('New_Password',256)) FROM
DUAL) = (SELECT LOWER(SHA('Confirm_Password',256)) FROM DUAL)
UPDATE
 CLIENT
SET
 PASS_WORD = (SELECT LOWER(SHA('new@stark@tony',256)) FROM DUAL)
WHERE
 CLIENT_ID = '43' ;
```

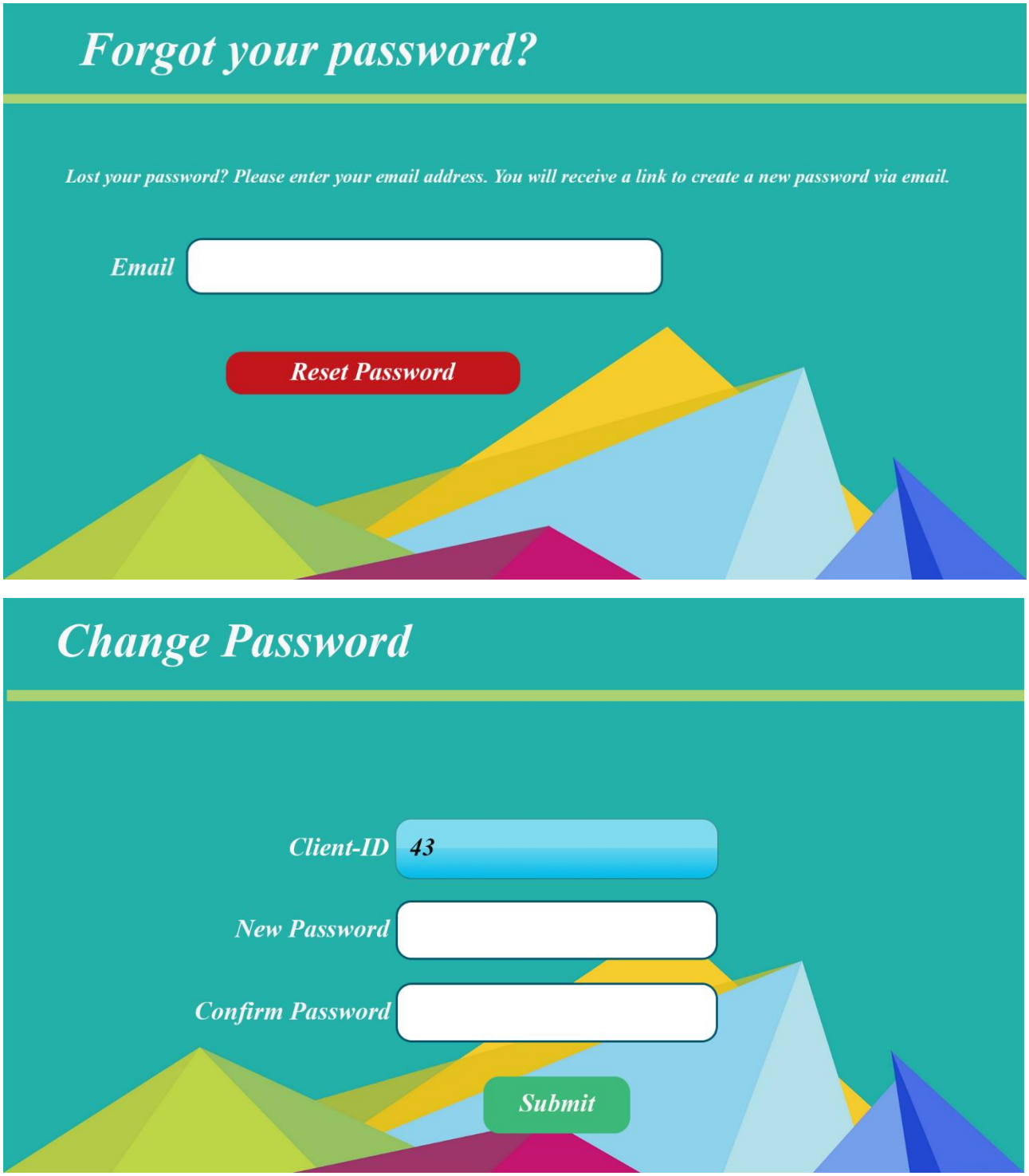


Figure 52 : Forgot Password

Change Password

SELECT * FROM DUAL WHERE (SELECT LOWER(SHA('New_Password',256)) FROM DUAL) = (SELECT LOWER(SHA('Confirm_Password',256)) FROM DUAL)

UPDATE

CLIENT

SET PASS_WORD = (SELECT LOWER(SHA('new@stark@tony',256)) FROM DUAL) WHERE

PASS_WORD = (SELECT LOWER(SHA('new@stark@tony',256)) FROM DUAL) AND CLIENT_ID = '43' ;

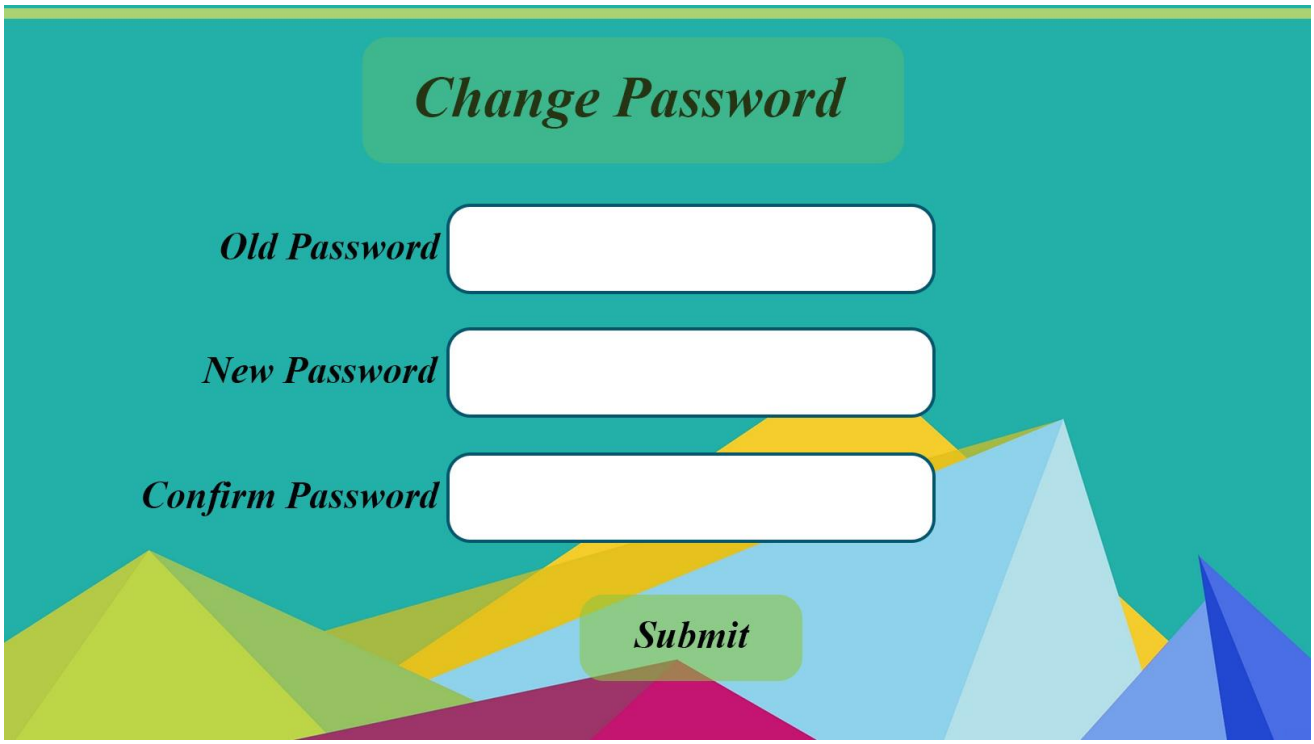


Figure 53 : Change Password

Client Update Profile

UPDATE

CLIENT

SET FIRST_NAME ='TONY',

LAST_NAME = 'STARK',

GENDER = 'M',

EMAIL = 'stark@stark.com',

STREET_ADDRESS1 = '',

STREET_ADDRESS2 = '',

CITY = '',

STATE = '',

COUNTRY = '',

ZIP = '',

```
 BIRTH_DATE = DATE '1996-03-26'
WHERE
CLIENT_ID = '43' ;
```

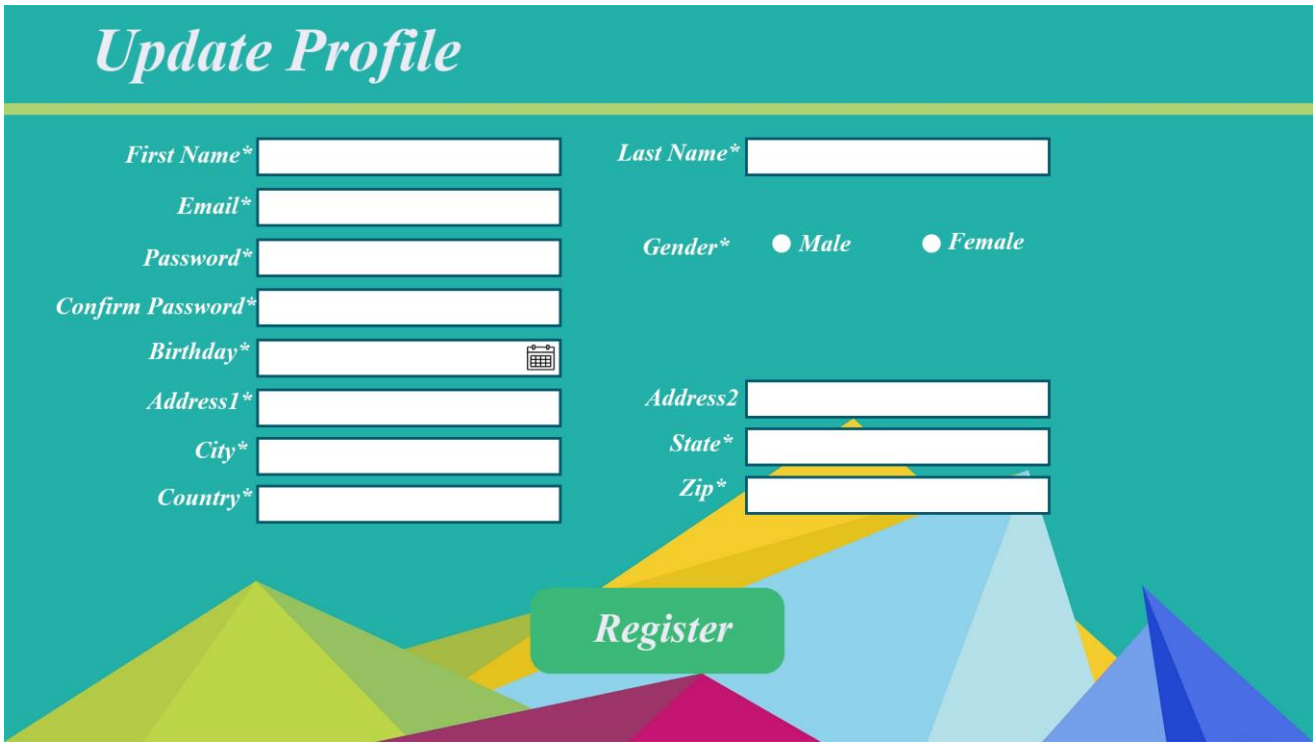


Figure 54 : Update Profile

Client Dashboard

- To display all the tickets created by a client having an id number 43 and the record should not be deleted. Also display according to the ticket priority and creation date.

```
  SELECT
  TICKET_MASTER.TICKET_ID,
  CLIENT.EMAIL,
  TICKET_ACTIVITY.SUBJECT,
  STAFF.USER_NAME,
  TICKET_MASTER.CREATED_DATE,
  TICKET_MASTER.TICKET_STATUS
FROM
  TICKET_MASTER
    INNER JOIN CLIENT
```

```
        ON TICKET_MASTER.CLIENT_ID = CLIENT.CLIENT_ID
    INNER JOIN TICKET_ACTIVITY
        ON TICKET_MASTER.TICKET_ID = TICKET_ACTIVITY.TICKET_ID
    INNER JOIN STAFF
        ON TICKET_MASTER.ASSIGN_TO = STAFF.STAFF_ID
WHERE
   TICKET_ACTIVITY.ACTIVITY_TYPE LIKE 'Create' AND
   TICKET_MASTER.CLIENT_ID = '43' AND
   TICKET_MASTER. IS_DELETED = 'N'
ORDER BY
   TICKET_MASTER.PRIORITY,
   TICKET_MASTER.CREATED_DATE;
```

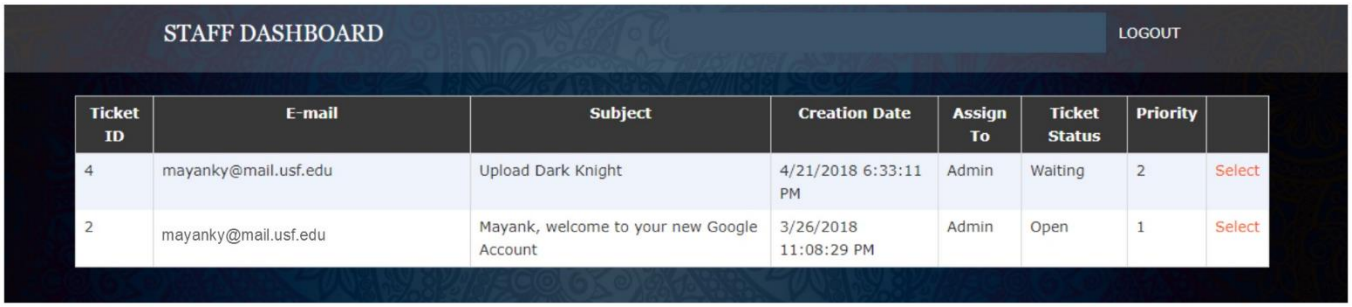


Figure 55 : Client Dashboard

Search in Dashboard

- Display tickets according to the term user uses to search in. For example, user wants to search for ticket whose client is 43, It should display only those set of tickets whose client id is 43.

```
SELECT
   TICKET_MASTER.TICKET_ID,
   CLIENT.EMAIL,
   TICKET_ACTIVITY.SUBJECT,
   STAFF.USER_NAME,
   TICKET_MASTER.CREATED_DATE,
   TICKET_MASTER.TICKET_STATUS
FROM
```

```
    TICKET_MASTER
      INNER JOIN CLIENT
        ON TICKET_MASTER.CLIENT_ID = CLIENT.CLIENT_ID
      INNER JOIN TICKET_ACTIVITY
        ON TICKET_MASTER.TICKET_ID = TICKET_ACTIVITY.TICKET_ID
      INNER JOIN STAFF
        ON TICKET_MASTER.ASSIGN_TO = STAFF.STAFF_ID
WHERE
    TICKET_ACTIVITY.ACTIVITY_TYPE LIKE 'Create' AND
    TICKET_MASTER.CLIENT_ID = ‘43’ AND
    TICKET_MASTER.IS_DELETED = 'N' AND (
    TICKET_MASTER.TICKET_ID LIKE '%43%' OR
    CLIENT.EMAIL LIKE '%43%' OR
    TICKET_ACTIVITY.SUBJECT LIKE '%43%' OR
    STAFF.USER_NAME  LIKE '%43%' OR
    TICKET_MASTER.CREATED_DATE  LIKE '%43%' OR
    TICKET_MASTER.TICKET_STATUS  LIKE '%43%'  OR
   TICKET_MASTER.TICKET_ID LIKE UPPER('%43%') OR
    CLIENT.EMAIL LIKE UPPER('%13%') OR
    TICKET_ACTIVITY.SUBJECT LIKE UPPER('%13%') OR
    STAFF.USER_NAME  LIKE UPPER('%43%') OR
    TICKET_MASTER.CREATED_DATE  LIKE UPPER('%43%') OR
    TICKET_MASTER.TICKET_STATUS  LIKE UPPER('%43%') OR
    TICKET_MASTER.TICKET_ID LIKE LOWER('%43%') OR
    CLIENT.EMAIL LIKE LOWER('%43%') OR
    TICKET_ACTIVITY.SUBJECT LIKE LOWER('%43%') OR
    STAFF.USER_NAME  LIKE LOWER('%43%') OR
    TICKET_MASTER.CREATED_DATE  LIKE LOWER('%43%') OR
    TICKET_MASTER.TICKET_STATUS  LIKE LOWER('%43%'))
ORDER BY
```

```
TICKET_MASTER.PRIORITY,
TICKET_MASTER.CREATED_DATE;
```

Create Ticket

- Insert a new record/create a ticket in the "Ticket_Master " table:

```
INSERT INTO TICKET_MASTER values(
Ticket_Master_SEQ.NEXTVAL,
'43',
CURRENT_TIMESTAMP,
'Web',
'Complain',",
'New',
2, ", ",
'N');
```

- Insert a record into table “Ticket_Activity” when a client files a complain.

```
INSERT INTO TICKET_Activity values(
Ticket_Activity_SEQ.NEXTVAL,
'21',
'Create',
'Complian about a product',
'I would like to Complian about a product - coconut hair oil',Current_Timestamp,
'C', ", 'N');
```

Client Ticket Sort

- Sort by Ticket_ID in Ascending Order

```
SELECT
   TICKET_MASTER.TICKET_ID,
   CLIENT.EMAIL,
   TICKET_ACTIVITY.SUBJECT,
   STAFF.USER_NAME,
```

```
   TICKET_MASTER.CREATED_DATE,
   TICKET_MASTER.TICKET_STATUS
FROM
   TICKET_MASTER
      INNER JOIN CLIENT
         ON TICKET_MASTER.CLIENT_ID = CLIENT.CLIENT_ID
      INNER JOIN TICKET_ACTIVITY
         ON TICKET_MASTER.TICKET_ID = TICKET_ACTIVITY.TICKET_ID
      INNER JOIN STAFF
         ON TICKET_MASTER.ASSIGN_TO = STAFF.STAFF_ID
WHERE
   TICKET_ACTIVITY.ACTIVITY_TYPE LIKE 'Create' AND
   TICKET_MASTER.CLIENT_ID = 21 AND
   TICKET_MASTER. IS_DELETED = ‘N’
ORDER BY
   TICKET_MASTER.TICKET_ID;
```

- Sort by Ticket_ID in Descending Order

```
SELECT
   TICKET_MASTER.TICKET_ID,
   CLIENT.EMAIL,
   TICKET_ACTIVITY.SUBJECT,
   STAFF.USER_NAME,
   TICKET_MASTER.CREATED_DATE,
   TICKET_MASTER.TICKET_STATUS
FROM
   TICKET_MASTER
      INNER JOIN CLIENT
         ON TICKET_MASTER.CLIENT_ID = CLIENT.CLIENT_ID
      INNER JOIN TICKET_ACTIVITY
```

```
        ON TICKET_MASTER.TICKET_ID = TICKET_ACTIVITY.TICKET_ID
    INNER JOIN STAFF
        ON TICKET_MASTER.ASSIGN_TO = STAFF.STAFF_ID
WHERE
  TICKET_ACTIVITY.ACTIVITY_TYPE LIKE 'Create' AND
  TICKET_MASTER.CLIENT_ID = 21 AND
  TICKET_MASTER. IS_DELETED = ‘N’
ORDER BY
  TICKET_MASTER.TICKET_ID DESC;
```

- Sort ticket by Subject in ascending order.

```
SELECT
  TICKET_MASTER.TICKET_ID,
  CLIENT.EMAIL,
  TICKET_ACTIVITY.SUBJECT,
  STAFF.USER_NAME,
  TICKET_MASTER.CREATED_DATE,
  TICKET_MASTER.TICKET_STATUS
FROM
  TICKET_MASTER
    INNER JOIN CLIENT
      ON TICKET_MASTER.CLIENT_ID = CLIENT.CLIENT_ID
    INNER JOIN TICKET_ACTIVITY
      ON TICKET_MASTER.TICKET_ID = TICKET_ACTIVITY.TICKET_ID
    INNER JOIN STAFF
      ON TICKET_MASTER.ASSIGN_TO = STAFF.STAFF_ID
WHERE
  TICKET_ACTIVITY.ACTIVITY_TYPE LIKE 'Create' AND
  TICKET_MASTER.CLIENT_ID = 21 AND
  TICKET_MASTER. IS_DELETED = ‘N’
```

```
ORDER BY
   TICKET_MASTER.SUBJECT;
```

- Sort ticket by Subject in Descending order.

```
SELECT
   TICKET_MASTER.TICKET_ID,
   CLIENT.EMAIL,
   TICKET_ACTIVITY.SUBJECT,
   STAFF.USER_NAME,
   TICKET_MASTER.CREATED_DATE,
   TICKET_MASTER.TICKET_STATUS
FROM
   TICKET_MASTER
      INNER JOIN CLIENT
         ON TICKET_MASTER.CLIENT_ID = CLIENT.CLIENT_ID
      INNER JOIN TICKET_ACTIVITY
         ON TICKET_MASTER.TICKET_ID = TICKET_ACTIVITY.TICKET_ID
      INNER JOIN STAFF
         ON TICKET_MASTER.ASSIGN_TO = STAFF.STAFF_ID
WHERE
   TICKET_ACTIVITY.ACTIVITY_TYPE LIKE 'Create' AND
   TICKET_MASTER.CLIENT_ID = 21 AND
   TICKET_MASTER. IS_DELETED = ‘N’
ORDER BY
   TICKET_MASTER.SUBJECT DESC;
```

- Sort by Creation Date in Ascending Order

```
SELECT
   TICKET_MASTER.TICKET_ID,
   CLIENT.EMAIL,
```

```
   TICKET_ACTIVITY.SUBJECT,
   STAFF.USER_NAME,
   TICKET_MASTER.CREATED_DATE,
   TICKET_MASTER.TICKET_STATUS
FROM
   TICKET_MASTER
      INNER JOIN CLIENT
         ON TICKET_MASTER.CLIENT_ID = CLIENT.CLIENT_ID
      INNER JOIN TICKET_ACTIVITY
         ON TICKET_MASTER.TICKET_ID = TICKET_ACTIVITY.TICKET_ID
      INNER JOIN STAFF
         ON TICKET_MASTER.ASSIGN_TO = STAFF.STAFF_ID
WHERE
   TICKET_ACTIVITY.ACTIVITY_TYPE LIKE 'Create' AND
   TICKET_MASTER.CLIENT_ID = 21 AND
   TICKET_MASTER. IS_DELETED = ‘N’
ORDER BY
   TICKET_MASTER. CREATED_DATE;
```

- Sort by Creation Date in Descending Order

```
SELECT
   TICKET_MASTER.TICKET_ID,
   CLIENT.EMAIL,
   TICKET_ACTIVITY.SUBJECT,
   STAFF.USER_NAME,
   TICKET_MASTER.CREATED_DATE,
   TICKET_MASTER.TICKET_STATUS
FROM
   TICKET_MASTER
      INNER JOIN CLIENT
```

```
    ON TICKET_MASTER.CLIENT_ID = CLIENT.CLIENT_ID
  INNER JOIN TICKET_ACTIVITY
    ON TICKET_MASTER.TICKET_ID = TICKET_ACTIVITY.TICKET_ID
  INNER JOIN STAFF
    ON TICKET_MASTER.ASSIGN_TO = STAFF.STAFF_ID
WHERE
  TICKET_ACTIVITY.ACTIVITY_TYPE LIKE 'Create' AND
  TICKET_MASTER.CLIENT_ID = 21 AND
  TICKET_MASTER. IS_DELETED = 'N'
ORDER BY
  TICKET_MASTER. CREATED_DATE DESC;
```

Specific Ticket

- Display all the details related to a Client's Ticket

```
SELECT
  TICKET_ID,
  FIRST_NAME,
  LAST_NAME,
  EMAIL,
  CREATED_DATE,
  TICKET_SOURCE,
  CATEGORIES,
  USER_NAME AS ASSIGN_TO,
  TICKET_STATUS,
  (CASE
    WHEN TICKET_MASTER.PRIORITY = 1
    THEN 'EMERGENCY'
    WHEN TICKET_MASTER.PRIORITY = 2
    THEN 'HIGH'
    WHEN TICKET_MASTER.PRIORITY = 3
```

```
    THEN 'NORMAL'
    WHEN TICKET_MASTER.PRIORITY > 3
    THEN 'LOW'
  END)PRIORITY
FROM
  TICKET_MASTER
  INNER JOIN CLIENT
   ON (SELECT CLIENT_ID FROM TICKET_MASTER WHERE TICKET_ID = 21) =
CLIENT.CLIENT_ID
  LEFT OUTER JOIN STAFF
   ON (SELECT ASSIGN_TO FROM TICKET_MASTER WHERE TICKET_ID = 21) =
STAFF.STAFF_ID
WHERE
  TICKET_ID = 21;
```

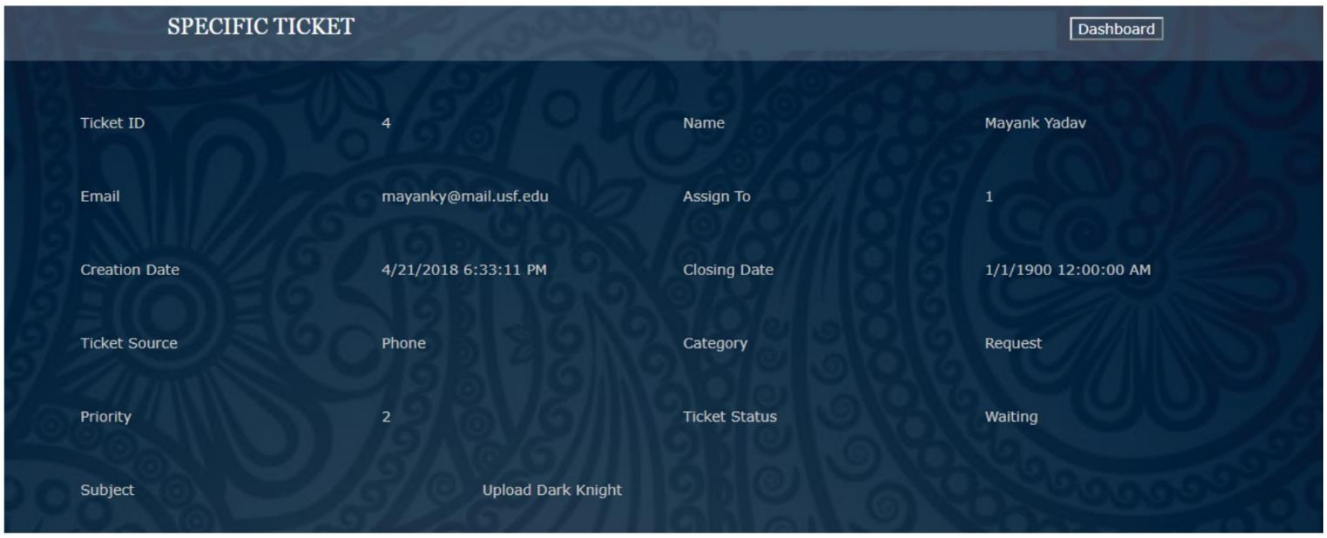


Figure 56 : Specific Ticket 1

- Display all the activities performed on a ticket

```
SELECT
  TICKET_ACTIVITY.ACTIVITY_TYPE,
  TICKET_ACTIVITY.ACTIVITY_DATE,
  (CASE
    WHEN TICKET_ACTIVITY.ACTED_BY = 'C'
    THEN (SELECT
```

```
        CLIENT.FIRST_NAME
      FROM
        CLIENT
        INNER JOIN TICKET_MASTER
        ON TICKET_MASTER.CLIENT_ID = CLIENT.CLIENT_ID
      WHERE
        TICKET_MASTER.TICKET_ID = 21)
    WHEN TICKET_ACTIVITY.ACTED_BY = 'S'
    THEN STAFF.USER_NAME
  END) ACTOR,
  TICKET_ACTIVITY.DESCRIPTION
FROM
  TICKET_ACTIVITY
  LEFT OUTER JOIN STAFF
    ON TICKET_ACTIVITY.ACTOR_ID = STAFF.STAFF_ID
WHERE
  TICKET_ACTIVITY.TICKET_ID = 21
ORDER BY
  TICKET_ACTIVITY.ACTIVITY_DATE;
```

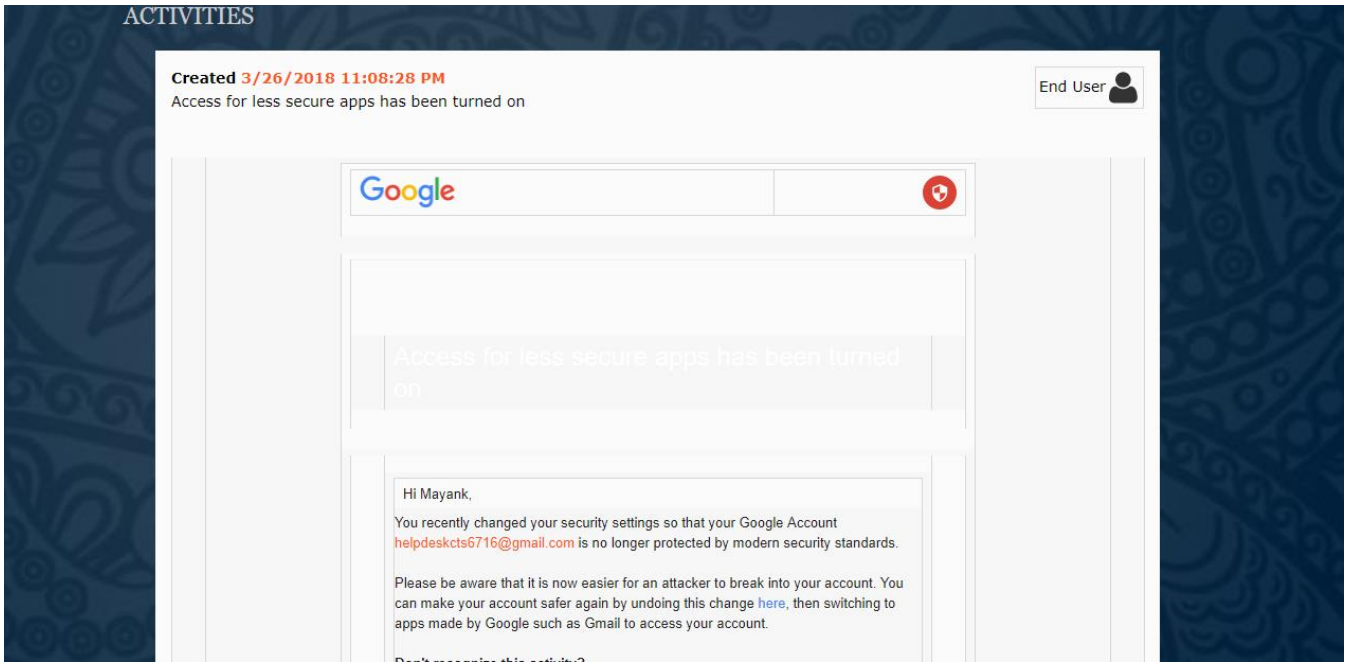


Figure 57 : Specific Ticket 2

Search in Ticket

- Display ticket activities according to the term user uses to search in. For example, user wants to search for an activity whose activity date is like 21 , then it should display only those set of tickets whose activity date is 21.

```
SELECT
  TICKET_ACTIVITY.ACTIVITY_TYPE,
  TICKET_ACTIVITY.ACTIVITY_DATE,
  (CASE
    WHEN TICKET_ACTIVITY.ACTED_BY = 'C'
    THEN (SELECT
        CLIENT.FIRST_NAME
      FROM
        CLIENT
        INNER JOIN TICKET_MASTER
        ON TICKET_MASTER.CLIENT_ID = CLIENT.CLIENT_ID
      WHERE
        TICKET_MASTER.TICKET_ID = 21)
    WHEN TICKET_ACTIVITY.ACTED_BY = 'S'
    THEN STAFF.USER_NAME
```

```
    END) ACTOR,
    TICKET_ACTIVITY.DESCRIPTION
FROM
    TICKET_ACTIVITY
    LEFT OUTER JOIN STAFF
        ON TICKET_ACTIVITY.ACTOR_ID = STAFF.STAFF_ID
WHERE
    TICKET_ACTIVITY.TICKET_ID = 21 AND (
    TICKET_ACTIVITY.ACTIVITY_TYPE LIKE '%21%' OR
    TICKET_ACTIVITY.ACTIVITY_DATE LIKE '%21%' OR
    STAFF.USER_NAME LIKE '%21%' OR
    TICKET_ACTIVITY.ACTED_BY LIKE '%21%' OR
    TICKET_ACTIVITY.DESCRIPTION LIKE '%21%' OR
    TICKET_ACTIVITY.ACTIVITY_TYPE LIKE UPPER('%21%') OR
    TICKET_ACTIVITY.ACTIVITY_DATE LIKE UPPER('%21%') OR
    STAFF.USER_NAME LIKE UPPER('%21%') OR
    TICKET_ACTIVITY.ACTED_BY LIKE UPPER('%21%') OR
    TICKET_ACTIVITY.DESCRIPTION LIKE UPPER('%21%') OR
    TICKET_ACTIVITY.ACTIVITY_TYPE LIKE LOWER('%21%') OR
    TICKET_ACTIVITY.ACTIVITY_DATE LIKE LOWER('%21%') OR
    STAFF.USER_NAME LIKE LOWER('%21%') OR
    TICKET_ACTIVITY.ACTED_BY LIKE LOWER('%21%') OR
    TICKET_ACTIVITY.DESCRIPTION LIKE LOWER('%21%'))
ORDER BY
    TICKET_ACTIVITY.ACTIVITY_DATE;
```

Post Customer Reply

- Insert a record in the “ticket_activity” table whenever a customer posts a reply in a Ticket

```
INSERT INTO TICKET_Activity values(
Ticket_Activity_SEQ.NEXTVAL,
```

'21',

'Reply',

'Reply from customer',

'I am still waiting for a reply’,

Current_Timestamp, 'C', '', 'N');

- Create an activity to store this event.

INSERT INTO TICKET_Activity values(

Ticket_Activity_SEQ.NEXTVAL,

'21',

'Ticket Status',

'Ticket Status has been changed',

'Ticket has been changed to waiting for customer response',

Current_Timestamp, 'C', '', 'N');

- Update the Ticket Status to Waiting

UPDATE TICKET_MASTER

SET TICKET_STATUS = 'Waiting'

WHERE TICKET_ID = '21';

Figure 58 : Post Reply

Delete Ticket

- Delete a Specific ticket

```
INSERT INTO TICKET_Activity values(
Ticket_Activity_SEQ.NEXTVAL,
'21',
'Ticket Deleted',
'Ticket has been deleted by the Client’,
'Ticket 21 has been deleted by the Client’,
Current_Timestamp, 'C', '', 'N');
```

- Update the Is_Deleted value to Y for Yes

```
UPDATE TICKET_MASTER
SET IS_DELETED = 'Y'
WHERE TICKET_ID = '21';
```

Staff Side

Staff Registration

```
INSERT
INTO
Staff
values(
Staff_SEQ.NEXTVAL,
'secondAdmin',
(SELECT LOWER(SHA(‘second@admin',256)) FROM DUAL),
'Y');
```

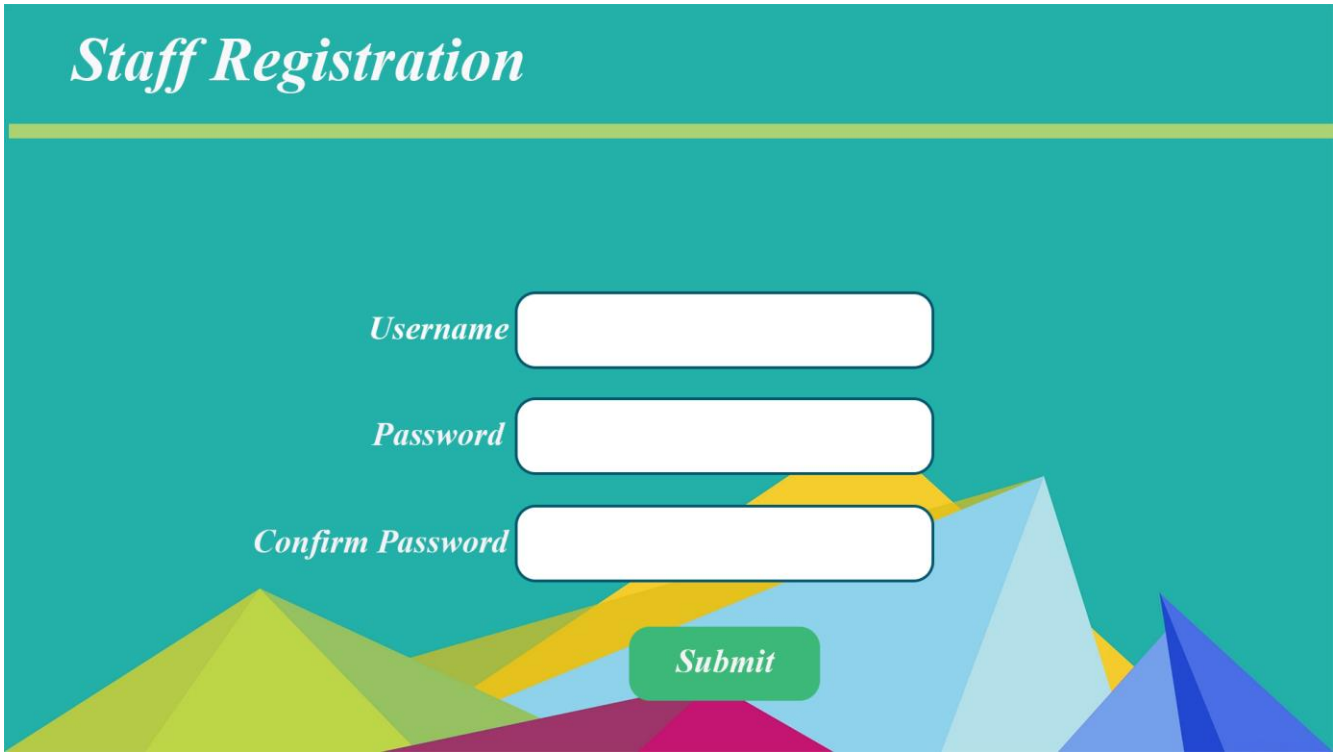


Figure 59 : Staff Registration

Staff Login

```
SELECT
USER_NAME,
PASS_WORD
FROM
STAFF
WHERE
USER_NAME = 'Admin' AND
PASS_WORD = (SELECT LOWER(SHA(‘second@admin',256)) FROM DUAL);
```

Figure 60 : Staff Registration

Staff Dashboard

- Show all Tickets to a Staff member

```
SELECT
   TICKET_MASTER.TICKET_ID,
   CLIENT.EMAIL,
   TICKET_ACTIVITY.SUBJECT,
   STAFF.USER_NAME,
   TICKET_MASTER.CREATED_DATE,
   TICKET_MASTER.TICKET_STATUS,
   (CASE
      WHEN TICKET_MASTER.PRIORITY = 1
      THEN 'EMERGENCY'
      WHEN TICKET_MASTER.PRIORITY = 2
      THEN 'HIGH'
      WHEN TICKET_MASTER.PRIORITY = 3
      THEN 'NORMAL'
      WHEN TICKET_MASTER.PRIORITY > 3
      THEN 'LOW'
```

```
    END)PRIORITY
FROM
    TICKET_MASTER
        INNER JOIN CLIENT
            ON TICKET_MASTER.CLIENT_ID = CLIENT.CLIENT_ID
        INNER JOIN TICKET_ACTIVITY
            ON TICKET_MASTER.TICKET_ID = TICKET_ACTIVITY.TICKET_ID
        LEFT OUTER JOIN STAFF
            ON TICKET_MASTER.ASSIGN_TO = STAFF.STAFF_ID
WHERE
    TICKET_ACTIVITY.ACTIVITY_TYPE LIKE 'Create' AND
    TICKET_MASTER.IS_DELETED = 'N'
ORDER BY
    TICKET_MASTER.PRIORITY,
    TICKET_MASTER.CREATED_DATE;
```

Ticket ID	E-mail	Subject	Creation Date	Assign To	Ticket Status	Priority	
8	rt@rt.com	lc vldf mv	4/22/2018 5:45:09 PM		New	2	Select
7	gh@gh.com	fjvbjfv	4/22/2018 5:40:37 PM		5	5	Select
6	mayanky@mail.usf.edu	Add Chris Pratt Movies	4/21/2018 6:36:28 PM		New	3	Select
5	mayanky@mail.usf.edu	Movie not Playing	4/21/2018 6:35:24 PM		Resolved	1	Select
4	mayanky@mail.usf.edu	Upload Dark Knight	4/21/2018 6:33:11 PM	Admin	Waiting	2	Select
1	Google <no-reply@accounts.google.com>	Access for less secure apps has been turned on	3/26/2018 11:08:27 PM		Open	1	Select

Figure 61 : Staff Dashboard

- Display only those tickets that assigned are assigned to a specific staff member.

```
SELECT
    TICKET_MASTER.TICKET_ID,
    CLIENT.EMAIL,
    TICKET_ACTIVITY.SUBJECT,
    STAFF.USER_NAME,
```

```
   TICKET_MASTER.CREATED_DATE,
   TICKET_MASTER.TICKET_STATUS,
   (CASE
      WHEN TICKET_MASTER.PRIORITY = 1
      THEN 'EMERGENCY'
      WHEN TICKET_MASTER.PRIORITY = 2
      THEN 'HIGH'
      WHEN TICKET_MASTER.PRIORITY = 3
      THEN 'NORMAL'
      WHEN TICKET_MASTER.PRIORITY > 3
      THEN 'LOW'
   END)PRIORITY
FROM
   TICKET_MASTER
      INNER JOIN CLIENT
         ON TICKET_MASTER.CLIENT_ID = CLIENT.CLIENT_ID
      INNER JOIN TICKET_ACTIVITY
         ON TICKET_MASTER.TICKET_ID = TICKET_ACTIVITY.TICKET_ID
      LEFT OUTER JOIN STAFF
         ON TICKET_MASTER.ASSIGN_TO = STAFF.STAFF_ID
WHERE
   TICKET_ACTIVITY.ACTIVITY_TYPE LIKE 'Create' AND
   TICKET_MASTER.IS_DELETED = 'N' AND
   TICKET_MASTER.ASSIGN_TO = ‘1’
ORDER BY
   TICKET_MASTER.PRIORITY,
   TICKET_MASTER.CREATED_DATE;
```

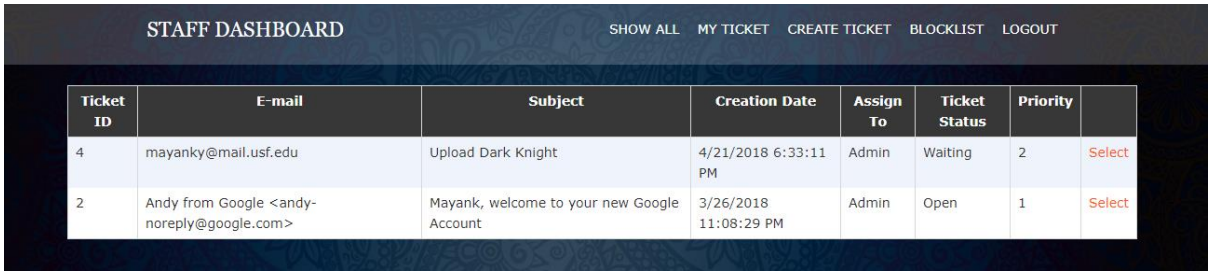


Figure 62 : Staff Dashboard Specific

- Show those tickets which are not assigned to anyone

```
SELECT
   TICKET_MASTER.TICKET_ID,
   CLIENT.EMAIL,
   TICKET_ACTIVITY.SUBJECT,
   STAFF.USER_NAME,
   TICKET_MASTER.CREATED_DATE,
   TICKET_MASTER.TICKET_STATUS,
   (CASE
      WHEN TICKET_MASTER.PRIORITY = 1
      THEN 'EMERGENCY'
      WHEN TICKET_MASTER.PRIORITY = 2
      THEN 'HIGH'
      WHEN TICKET_MASTER.PRIORITY = 3
      THEN 'NORMAL'
      WHEN TICKET_MASTER.PRIORITY > 3
      THEN 'LOW'
   END)PRIORITY
FROM
   TICKET_MASTER
      INNER JOIN CLIENT
         ON TICKET_MASTER.CLIENT_ID = CLIENT.CLIENT_ID
      INNER JOIN TICKET_ACTIVITY
         ON TICKET_MASTER.TICKET_ID = TICKET_ACTIVITY.TICKET_ID
      LEFT OUTER JOIN STAFF
```

```
        ON TICKET_MASTER.ASSIGN_TO = STAFF.STAFF_ID
WHERE
   TICKET_ACTIVITY.ACTIVITY_TYPE LIKE 'Create' AND
   TICKET_MASTER.IS_DELETED = 'N' AND
   TICKET_MASTER.ASSIGN_TO IS NULL
ORDER BY
   TICKET_MASTER.PRIORITY,
   TICKET_MASTER.CREATED_DATE;
```

Assign a ticket to itself

- Allow the staff to assign a ticket to himself by specifying the client id and ticket id he wants to work with.

```
INSERT
INTO
TICKET_Activity
values(
Ticket_Activity_SEQ.NEXTVAL,
'22',
'Assign',
'Staff has been assigned to your ticket',
'Peter (21) has been assigned to your ticket',
Current_Timestamp,
'S', '21', 'N');


UPDATE
TICKET_MASTER
SET
ASSIGN_TO = '21'
WHERE TICKET_ID = '22';
```

Figure 63 : Assign Ticket

Staff Ticket Sort

- Sort by Ticket_ID in ascending order

```
SELECT
   TICKET_MASTER.TICKET_ID,
   CLIENT.EMAIL,
   TICKET_ACTIVITY.SUBJECT,
   STAFF.USER_NAME,
   TICKET_MASTER.CREATED_DATE,
   TICKET_MASTER.TICKET_STATUS,
   (CASE
      WHEN TICKET_MASTER.PRIORITY = 1
      THEN 'EMERGENCY'
      WHEN TICKET_MASTER.PRIORITY = 2
      THEN 'HIGH'
      WHEN TICKET_MASTER.PRIORITY = 3
      THEN 'NORMAL'
      WHEN TICKET_MASTER.PRIORITY > 3
      THEN 'LOW'
```

```
   END)PRIORITY
FROM
   TICKET_MASTER
      INNER JOIN CLIENT
         ON TICKET_MASTER.CLIENT_ID = CLIENT.CLIENT_ID
      INNER JOIN TICKET_ACTIVITY
         ON TICKET_MASTER.TICKET_ID = TICKET_ACTIVITY.TICKET_ID
      LEFT OUTER JOIN STAFF
         ON TICKET_MASTER.ASSIGN_TO = STAFF.STAFF_ID
WHERE
   TICKET_ACTIVITY.ACTIVITY_TYPE LIKE 'Create' AND
   TICKET_MASTER.IS_DELETED = 'N' AND
   TICKET_MASTER.ASSIGN_TO IS NULL
ORDER BY
   TICKET_MASTER.TICKET_ID;
```

- Sort by Ticket_ID in descending order

```
SELECT
   TICKET_MASTER.TICKET_ID,
   CLIENT.EMAIL,
   TICKET_ACTIVITY.SUBJECT,
   STAFF.USER_NAME,
   TICKET_MASTER.CREATED_DATE,
   TICKET_MASTER.TICKET_STATUS,
   (CASE
      WHEN TICKET_MASTER.PRIORITY = 1
      THEN 'EMERGENCY'
      WHEN TICKET_MASTER.PRIORITY = 2
      THEN 'HIGH'
      WHEN TICKET_MASTER.PRIORITY = 3
```

```
      THEN 'NORMAL'
      WHEN TICKET_MASTER.PRIORITY > 3
      THEN 'LOW'
   END)PRIORITY
FROM
   TICKET_MASTER
      INNER JOIN CLIENT
         ON TICKET_MASTER.CLIENT_ID = CLIENT.CLIENT_ID
      INNER JOIN TICKET_ACTIVITY
         ON TICKET_MASTER.TICKET_ID = TICKET_ACTIVITY.TICKET_ID
      LEFT OUTER JOIN STAFF
         ON TICKET_MASTER.ASSIGN_TO = STAFF.STAFF_ID
WHERE
   TICKET_ACTIVITY.ACTIVITY_TYPE LIKE 'Create' AND
   TICKET_MASTER.IS_DELETED = 'N' AND
   TICKET_MASTER.ASSIGN_TO IS NULL
ORDER BY
   TICKET_MASTER.TICKET_ID DESC;
```

- Sort by Email id in ascending order

```
SELECT
   TICKET_MASTER.TICKET_ID,
   CLIENT.EMAIL,
   TICKET_ACTIVITY.SUBJECT,
   STAFF.USER_NAME,
   TICKET_MASTER.CREATED_DATE,
   TICKET_MASTER.TICKET_STATUS,
   (CASE
      WHEN TICKET_MASTER.PRIORITY = 1
      THEN 'EMERGENCY'
```

```
      WHEN TICKET_MASTER.PRIORITY = 2
      THEN 'HIGH'
      WHEN TICKET_MASTER.PRIORITY = 3
      THEN 'NORMAL'
      WHEN TICKET_MASTER.PRIORITY > 3
      THEN 'LOW'
   END)PRIORITY
FROM
   TICKET_MASTER
      INNER JOIN CLIENT
         ON TICKET_MASTER.CLIENT_ID = CLIENT.CLIENT_ID
      INNER JOIN TICKET_ACTIVITY
         ON TICKET_MASTER.TICKET_ID = TICKET_ACTIVITY.TICKET_ID
      LEFT OUTER JOIN STAFF
         ON TICKET_MASTER.ASSIGN_TO = STAFF.STAFF_ID
WHERE
   TICKET_ACTIVITY.ACTIVITY_TYPE LIKE 'Create' AND
   TICKET_MASTER.IS_DELETED = 'N' AND
   TICKET_MASTER.ASSIGN_TO IS NULL
ORDER BY
   TICKET_MASTER.EMAIL;
```

- Sort by Email in descending order

```
SELECT
   TICKET_MASTER.TICKET_ID,
   CLIENT.EMAIL,
   TICKET_ACTIVITY.SUBJECT,
   STAFF.USER_NAME,
   TICKET_MASTER.CREATED_DATE,
   TICKET_MASTER.TICKET_STATUS,
```

```
   (CASE
      WHEN TICKET_MASTER.PRIORITY = 1
      THEN 'EMERGENCY'
      WHEN TICKET_MASTER.PRIORITY = 2
      THEN 'HIGH'
      WHEN TICKET_MASTER.PRIORITY = 3
      THEN 'NORMAL'
      WHEN TICKET_MASTER.PRIORITY > 3
      THEN 'LOW'
   END)PRIORITY
FROM
   TICKET_MASTER
      INNER JOIN CLIENT
         ON TICKET_MASTER.CLIENT_ID = CLIENT.CLIENT_ID
      INNER JOIN TICKET_ACTIVITY
         ON TICKET_MASTER.TICKET_ID = TICKET_ACTIVITY.TICKET_ID
      LEFT OUTER JOIN STAFF
         ON TICKET_MASTER.ASSIGN_TO = STAFF.STAFF_ID
WHERE
   TICKET_ACTIVITY.ACTIVITY_TYPE LIKE 'Create' AND
   TICKET_MASTER.IS_DELETED = 'N' AND
   TICKET_MASTER.ASSIGN_TO IS NULL
ORDER BY
   TICKET_MASTER.EMAIL DESC;
```

- Sort by Subject in ascending order

```
SELECT
   TICKET_MASTER.TICKET_ID,
   CLIENT.EMAIL,
   TICKET_ACTIVITY.SUBJECT,
```

```
    STAFF.USER_NAME,
    TICKET_MASTER.CREATED_DATE,
    TICKET_MASTER.TICKET_STATUS,
    (CASE
        WHEN TICKET_MASTER.PRIORITY = 1
        THEN 'EMERGENCY'
        WHEN TICKET_MASTER.PRIORITY = 2
        THEN 'HIGH'
        WHEN TICKET_MASTER.PRIORITY = 3
        THEN 'NORMAL'
        WHEN TICKET_MASTER.PRIORITY > 3
        THEN 'LOW'
    END)PRIORITY
FROM
    TICKET_MASTER
        INNER JOIN CLIENT
            ON TICKET_MASTER.CLIENT_ID = CLIENT.CLIENT_ID
        INNER JOIN TICKET_ACTIVITY
            ON TICKET_MASTER.TICKET_ID = TICKET_ACTIVITY.TICKET_ID
        LEFT OUTER JOIN STAFF
            ON TICKET_MASTER.ASSIGN_TO = STAFF.STAFF_ID
WHERE
    TICKET_ACTIVITY.ACTIVITY_TYPE LIKE 'Create' AND
    TICKET_MASTER.IS_DELETED = 'N' AND
    TICKET_MASTER.ASSIGN_TO IS NULL
ORDER BY
    TICKET_MASTER.SUBJECT;
```

- Sort by Subject in descending order

```
SELECT
```

```
   TICKET_MASTER.TICKET_ID,
   CLIENT.EMAIL,
   TICKET_ACTIVITY.SUBJECT,
   STAFF.USER_NAME,
   TICKET_MASTER.CREATED_DATE,
   TICKET_MASTER.TICKET_STATUS,
   (CASE
      WHEN TICKET_MASTER.PRIORITY = 1
      THEN 'EMERGENCY'
      WHEN TICKET_MASTER.PRIORITY = 2
      THEN 'HIGH'
      WHEN TICKET_MASTER.PRIORITY = 3
      THEN 'NORMAL'
      WHEN TICKET_MASTER.PRIORITY > 3
      THEN 'LOW'
   END)PRIORITY
FROM
   TICKET_MASTER
      INNER JOIN CLIENT
         ON TICKET_MASTER.CLIENT_ID = CLIENT.CLIENT_ID
      INNER JOIN TICKET_ACTIVITY
         ON TICKET_MASTER.TICKET_ID = TICKET_ACTIVITY.TICKET_ID
      LEFT OUTER JOIN STAFF
         ON TICKET_MASTER.ASSIGN_TO = STAFF.STAFF_ID
WHERE
   TICKET_ACTIVITY.ACTIVITY_TYPE LIKE 'Create' AND
   TICKET_MASTER.IS_DELETED = 'N' AND
   TICKET_MASTER.ASSIGN_TO IS NULL
ORDER BY
   TICKET_MASTER.SUBJECT DESC;
```

- Sort by Status in ascending order

```
SELECT
   TICKET_MASTER.TICKET_ID,
   CLIENT.EMAIL,
   TICKET_ACTIVITY.SUBJECT,
   STAFF.USER_NAME,
   TICKET_MASTER.CREATED_DATE,
   TICKET_MASTER.TICKET_STATUS,
   (CASE
      WHEN TICKET_MASTER.PRIORITY = 1
      THEN 'EMERGENCY'
      WHEN TICKET_MASTER.PRIORITY = 2
      THEN 'HIGH'
      WHEN TICKET_MASTER.PRIORITY = 3
      THEN 'NORMAL'
      WHEN TICKET_MASTER.PRIORITY > 3
      THEN 'LOW'
   END)PRIORITY
FROM
   TICKET_MASTER
      INNER JOIN CLIENT
         ON TICKET_MASTER.CLIENT_ID = CLIENT.CLIENT_ID
      INNER JOIN TICKET_ACTIVITY
         ON TICKET_MASTER.TICKET_ID = TICKET_ACTIVITY.TICKET_ID
      LEFT OUTER JOIN STAFF
         ON TICKET_MASTER.ASSIGN_TO = STAFF.STAFF_ID
WHERE
   TICKET_ACTIVITY.ACTIVITY_TYPE LIKE 'Create' AND
   TICKET_MASTER.IS_DELETED = 'N' AND
```

```
   TICKET_MASTER.ASSIGN_TO IS NULL
ORDER BY
   TICKET_MASTER.TICKET_STATUS;
```

- Sort by Status in descending order

```
SELECT
   TICKET_MASTER.TICKET_ID,
   CLIENT.EMAIL,
   TICKET_ACTIVITY.SUBJECT,
   STAFF.USER_NAME,
   TICKET_MASTER.CREATED_DATE,
   TICKET_MASTER.TICKET_STATUS,
   (CASE
      WHEN TICKET_MASTER.PRIORITY = 1
      THEN 'EMERGENCY'
      WHEN TICKET_MASTER.PRIORITY = 2
      THEN 'HIGH'
      WHEN TICKET_MASTER.PRIORITY = 3
      THEN 'NORMAL'
      WHEN TICKET_MASTER.PRIORITY > 3
      THEN 'LOW'
   END)PRIORITY
FROM
   TICKET_MASTER
      INNER JOIN CLIENT
         ON TICKET_MASTER.CLIENT_ID = CLIENT.CLIENT_ID
      INNER JOIN TICKET_ACTIVITY
         ON TICKET_MASTER.TICKET_ID = TICKET_ACTIVITY.TICKET_ID
      LEFT OUTER JOIN STAFF
         ON TICKET_MASTER.ASSIGN_TO = STAFF.STAFF_ID
```

```
WHERE
   TICKET_ACTIVITY.ACTIVITY_TYPE LIKE 'Create' AND
   TICKET_MASTER.IS_DELETED = 'N' AND
   TICKET_MASTER.ASSIGN_TO IS NULL
ORDER BY
   TICKET_MASTER.TICKET_STATUS DESC;
```

- Sort by Priority in ascending order

```
SELECT
   TICKET_MASTER.TICKET_ID,
   CLIENT.EMAIL,
   TICKET_ACTIVITY.SUBJECT,
   STAFF.USER_NAME,
   TICKET_MASTER.CREATED_DATE,
   TICKET_MASTER.TICKET_STATUS,
   (CASE
      WHEN TICKET_MASTER.PRIORITY = 1
      THEN 'EMERGENCY'
      WHEN TICKET_MASTER.PRIORITY = 2
      THEN 'HIGH'
      WHEN TICKET_MASTER.PRIORITY = 3
      THEN 'NORMAL'
      WHEN TICKET_MASTER.PRIORITY > 3
      THEN 'LOW'
   END)PRIORITY
FROM
   TICKET_MASTER
      INNER JOIN CLIENT
         ON TICKET_MASTER.CLIENT_ID = CLIENT.CLIENT_ID
      INNER JOIN TICKET_ACTIVITY
```

```
        ON TICKET_MASTER.TICKET_ID = TICKET_ACTIVITY.TICKET_ID
    LEFT OUTER JOIN STAFF
        ON TICKET_MASTER.ASSIGN_TO = STAFF.STAFF_ID
WHERE
  TICKET_ACTIVITY.ACTIVITY_TYPE LIKE 'Create' AND
  TICKET_MASTER.IS_DELETED = 'N' AND
  TICKET_MASTER.ASSIGN_TO IS NULL
ORDER BY
  TICKET_MASTER.PRIORITY;
```

- Sort by Priority in descending order

```
SELECT
  TICKET_MASTER.TICKET_ID,
  CLIENT.EMAIL,
  TICKET_ACTIVITY.SUBJECT,
  STAFF.USER_NAME,
  TICKET_MASTER.CREATED_DATE,
  TICKET_MASTER.TICKET_STATUS,
  (CASE
    WHEN TICKET_MASTER.PRIORITY = 1
    THEN 'EMERGENCY'
    WHEN TICKET_MASTER.PRIORITY = 2
    THEN 'HIGH'
    WHEN TICKET_MASTER.PRIORITY = 3
    THEN 'NORMAL'
    WHEN TICKET_MASTER.PRIORITY > 3
    THEN 'LOW'
  END)PRIORITY
FROM
  TICKET_MASTER
```

```
    INNER JOIN CLIENT
      ON TICKET_MASTER.CLIENT_ID = CLIENT.CLIENT_ID
    INNER JOIN TICKET_ACTIVITY
      ON TICKET_MASTER.TICKET_ID = TICKET_ACTIVITY.TICKET_ID
    LEFT OUTER JOIN STAFF
      ON TICKET_MASTER.ASSIGN_TO = STAFF.STAFF_ID
WHERE
  TICKET_ACTIVITY.ACTIVITY_TYPE LIKE 'Create' AND
  TICKET_MASTER.IS_DELETED = 'N' AND
  TICKET_MASTER.ASSIGN_TO IS NULL
ORDER BY
  TICKET_MASTER.PRIORITY DESC;
```

- Sort by Created_Date in ascending order

```
SELECT
  TICKET_MASTER.TICKET_ID,
  CLIENT.EMAIL,
  TICKET_ACTIVITY.SUBJECT,
  STAFF.USER_NAME,
  TICKET_MASTER.CREATED_DATE,
  TICKET_MASTER.TICKET_STATUS,
  (CASE
    WHEN TICKET_MASTER.PRIORITY = 1
    THEN 'EMERGENCY'
    WHEN TICKET_MASTER.PRIORITY = 2
    THEN 'HIGH'
    WHEN TICKET_MASTER.PRIORITY = 3
    THEN 'NORMAL'
    WHEN TICKET_MASTER.PRIORITY > 3
    THEN 'LOW'
```

```
   END)PRIORITY
FROM
   TICKET_MASTER
      INNER JOIN CLIENT
         ON TICKET_MASTER.CLIENT_ID = CLIENT.CLIENT_ID
      INNER JOIN TICKET_ACTIVITY
         ON TICKET_MASTER.TICKET_ID = TICKET_ACTIVITY.TICKET_ID
      LEFT OUTER JOIN STAFF
         ON TICKET_MASTER.ASSIGN_TO = STAFF.STAFF_ID
WHERE
   TICKET_ACTIVITY.ACTIVITY_TYPE LIKE 'Create' AND
   TICKET_MASTER.IS_DELETED = 'N' AND
   TICKET_MASTER.ASSIGN_TO IS NULL
ORDER BY
   TICKET_MASTER. Created_Date;
```

- Sort by Created_Date in descending order

```
SELECT
   TICKET_MASTER.TICKET_ID,
   CLIENT.EMAIL,
   TICKET_ACTIVITY.SUBJECT,
   STAFF.USER_NAME,
   TICKET_MASTER.CREATED_DATE,
   TICKET_MASTER.TICKET_STATUS,
   (CASE
      WHEN TICKET_MASTER.PRIORITY = 1
      THEN 'EMERGENCY'
      WHEN TICKET_MASTER.PRIORITY = 2
      THEN 'HIGH'
      WHEN TICKET_MASTER.PRIORITY = 3
```

```
      THEN 'NORMAL'
      WHEN TICKET_MASTER.PRIORITY > 3
      THEN 'LOW'
   END)PRIORITY
FROM
   TICKET_MASTER
      INNER JOIN CLIENT
         ON TICKET_MASTER.CLIENT_ID = CLIENT.CLIENT_ID
      INNER JOIN TICKET_ACTIVITY
         ON TICKET_MASTER.TICKET_ID = TICKET_ACTIVITY.TICKET_ID
      LEFT OUTER JOIN STAFF
         ON TICKET_MASTER.ASSIGN_TO = STAFF.STAFF_ID
WHERE
   TICKET_ACTIVITY.ACTIVITY_TYPE LIKE 'Create' AND
   TICKET_MASTER.IS_DELETED = 'N' AND
   TICKET_MASTER.ASSIGN_TO IS NULL
ORDER BY
   TICKET_MASTER. Created_Date DESC;
```

- Sort by Staff in ascending order

```
SELECT
   TICKET_MASTER.TICKET_ID,
   CLIENT.EMAIL,
   TICKET_ACTIVITY.SUBJECT,
   STAFF.USER_NAME,
   TICKET_MASTER.CREATED_DATE,
   TICKET_MASTER.TICKET_STATUS,
   (CASE
      WHEN TICKET_MASTER.PRIORITY = 1
      THEN 'EMERGENCY'
```

```
      WHEN TICKET_MASTER.PRIORITY = 2
      THEN 'HIGH'
      WHEN TICKET_MASTER.PRIORITY = 3
      THEN 'NORMAL'
      WHEN TICKET_MASTER.PRIORITY > 3
      THEN 'LOW'
   END)PRIORITY
FROM
   TICKET_MASTER
      INNER JOIN CLIENT
         ON TICKET_MASTER.CLIENT_ID = CLIENT.CLIENT_ID
      INNER JOIN TICKET_ACTIVITY
         ON TICKET_MASTER.TICKET_ID = TICKET_ACTIVITY.TICKET_ID
      LEFT OUTER JOIN STAFF
         ON TICKET_MASTER.ASSIGN_TO = STAFF.STAFF_ID
WHERE
   TICKET_ACTIVITY.ACTIVITY_TYPE LIKE 'Create' AND
   TICKET_MASTER.IS_DELETED = 'N' AND
   TICKET_MASTER.ASSIGN_TO IS NULL
ORDER BY
   STAFF.USER_NAME;
```

- Sort by Staff in descending order

```
SELECT
   TICKET_MASTER.TICKET_ID,
   CLIENT.EMAIL,
   TICKET_ACTIVITY.SUBJECT,
   STAFF.USER_NAME,
   TICKET_MASTER.CREATED_DATE,
   TICKET_MASTER.TICKET_STATUS,
```

```
   (CASE
      WHEN TICKET_MASTER.PRIORITY = 1
      THEN 'EMERGENCY'
      WHEN TICKET_MASTER.PRIORITY = 2
      THEN 'HIGH'
      WHEN TICKET_MASTER.PRIORITY = 3
      THEN 'NORMAL'
      WHEN TICKET_MASTER.PRIORITY > 3
      THEN 'LOW'
   END)PRIORITY
FROM
   TICKET_MASTER
      INNER JOIN CLIENT
         ON TICKET_MASTER.CLIENT_ID = CLIENT.CLIENT_ID
      INNER JOIN TICKET_ACTIVITY
         ON TICKET_MASTER.TICKET_ID = TICKET_ACTIVITY.TICKET_ID
      LEFT OUTER JOIN STAFF
         ON TICKET_MASTER.ASSIGN_TO = STAFF.STAFF_ID
WHERE
   TICKET_ACTIVITY.ACTIVITY_TYPE LIKE 'Create' AND
   TICKET_MASTER.IS_DELETED = 'N' AND
   TICKET_MASTER.ASSIGN_TO IS NULL
ORDER BY
      STAFF.USER_NAME DESC;
```

Search in Dashboard

- Display only those tickets for which the user has searched in.

```
SELECT
   TICKET_MASTER.TICKET_ID,
   CLIENT.EMAIL,
```

```
   TICKET_ACTIVITY.SUBJECT,
   STAFF.USER_NAME,
   TICKET_MASTER.CREATED_DATE,
   TICKET_MASTER.TICKET_STATUS,
   (CASE
      WHEN TICKET_MASTER.PRIORITY = 1
      THEN 'EMERGENCY'
      WHEN TICKET_MASTER.PRIORITY = 2
      THEN 'HIGH'
      WHEN TICKET_MASTER.PRIORITY = 3
      THEN 'NORMAL'
      WHEN TICKET_MASTER.PRIORITY > 3
      THEN 'LOW'
   END)PRIORITY
FROM
   TICKET_MASTER
      INNER JOIN CLIENT
         ON TICKET_MASTER.CLIENT_ID = CLIENT.CLIENT_ID
      INNER JOIN TICKET_ACTIVITY
         ON TICKET_MASTER.TICKET_ID = TICKET_ACTIVITY.TICKET_ID
      LEFT OUTER JOIN STAFF
         ON TICKET_MASTER.ASSIGN_TO = STAFF.STAFF_ID
WHERE
   TICKET_ACTIVITY.ACTIVITY_TYPE LIKE 'Create' AND
   TICKET_MASTER.IS_DELETED = 'N' AND (
   TICKET_MASTER.TICKET_ID LIKE '%21%' OR
   CLIENT.EMAIL LIKE '%21%' OR
   TICKET_ACTIVITY.SUBJECT LIKE '%21%' OR
   STAFF.USER_NAME  LIKE '%21%' OR
   TICKET_MASTER.CREATED_DATE  LIKE '%21%' OR
```

```
    TICKET_MASTER.TICKET_STATUS  LIKE '%21%' OR
    TICKET_MASTER.PRIORITY LIKE '%21%' OR
    TICKET_MASTER.TICKET_ID LIKE UPPER('%21%') OR
    CLIENT.EMAIL LIKE UPPER('%21%') OR
    TICKET_ACTIVITY.SUBJECT LIKE UPPER('%21%') OR
    STAFF.USER_NAME  LIKE UPPER('%21%') OR
    TICKET_MASTER.CREATED_DATE  LIKE UPPER('%21%') OR
    TICKET_MASTER.TICKET_STATUS  LIKE UPPER('%21%') OR
    TICKET_MASTER.PRIORITY LIKE UPPER('%21%') OR
    TICKET_MASTER.TICKET_ID LIKE LOWER('%21%') OR
    CLIENT.EMAIL LIKE LOWER('%21%') OR
    TICKET_ACTIVITY.SUBJECT LIKE LOWER('%21%') OR
    STAFF.USER_NAME  LIKE LOWER('%21%') OR
    TICKET_MASTER.CREATED_DATE  LIKE LOWER('%21%') OR
    TICKET_MASTER.TICKET_STATUS  LIKE LOWER('%21%') OR
    TICKET_MASTER.PRIORITY LIKE LOWER('%21%'))
ORDER BY
    TICKET_MASTER.PRIORITY,
    TICKET_MASTER.CREATED_DATE;
```

Specific Ticket

- Describe all details related to a particular ticket.

```
SELECT
    TICKET_ID,
    FIRST_NAME,
    LAST_NAME,
    EMAIL,
    CREATED_DATE,
    TICKET_SOURCE,
    CATEGORIES,
```

```
   USER_NAME AS ASSIGN_TO,
   TICKET_STATUS,
   (CASE
      WHEN TICKET_MASTER.PRIORITY = 1
      THEN 'EMERGENCY'
      WHEN TICKET_MASTER.PRIORITY = 2
      THEN 'HIGH'
      WHEN TICKET_MASTER.PRIORITY = 3
      THEN 'NORMAL'
      WHEN TICKET_MASTER.PRIORITY > 3
      THEN 'LOW'
   END)PRIORITY
FROM
   TICKET_MASTER
   INNER JOIN CLIENT
    ON (SELECT CLIENT_ID FROM TICKET_MASTER WHERE TICKET_ID = 21) =
CLIENT.CLIENT_ID
   LEFT OUTER JOIN STAFF
    ON (SELECT ASSIGN_TO FROM TICKET_MASTER WHERE TICKET_ID = 21) =
STAFF.STAFF_ID
WHERE
   TICKET_ID = 21;
```

- If the Closed date is NULL display the most recent activity date

```
SELECT
*
FROM (
SELECT ACTIVITY_ID, TICKET_ID, ACTIVITY_TYPE, ACTIVITY_DATE
 FROM TICKET_ACTIVITY
WHERE TICKET_ACTIVITY.TICKET_ID = '21'
ORDER BY TICKET_ACTIVITY.ACTIVITY_DATE DESC)
```

```
WHERE ROWNUM =1;
```

- Display all activities performed on a specific ticket

```
SELECT
   TICKET_ACTIVITY.ACTIVITY_TYPE,
   TICKET_ACTIVITY.ACTIVITY_DATE,
   (CASE
      WHEN TICKET_ACTIVITY.ACTED_BY = 'C'
      THEN (SELECT
            CLIENT.FIRST_NAME
          FROM
            CLIENT
            INNER JOIN TICKET_MASTER
            ON TICKET_MASTER.CLIENT_ID = CLIENT.CLIENT_ID
         WHERE
            TICKET_MASTER.TICKET_ID = 21)
      WHEN TICKET_ACTIVITY.ACTED_BY = 'S'
      THEN STAFF.USER_NAME
   END) ACTOR,
   TICKET_ACTIVITY.DESCRIPTION
FROM
   TICKET_ACTIVITY
   LEFT OUTER JOIN STAFF
      ON TICKET_ACTIVITY.ACTOR_ID = STAFF.STAFF_ID
WHERE
   TICKET_ACTIVITY.TICKET_ID = 21
ORDER BY
   TICKET_ACTIVITY.ACTIVITY_DATE;
```

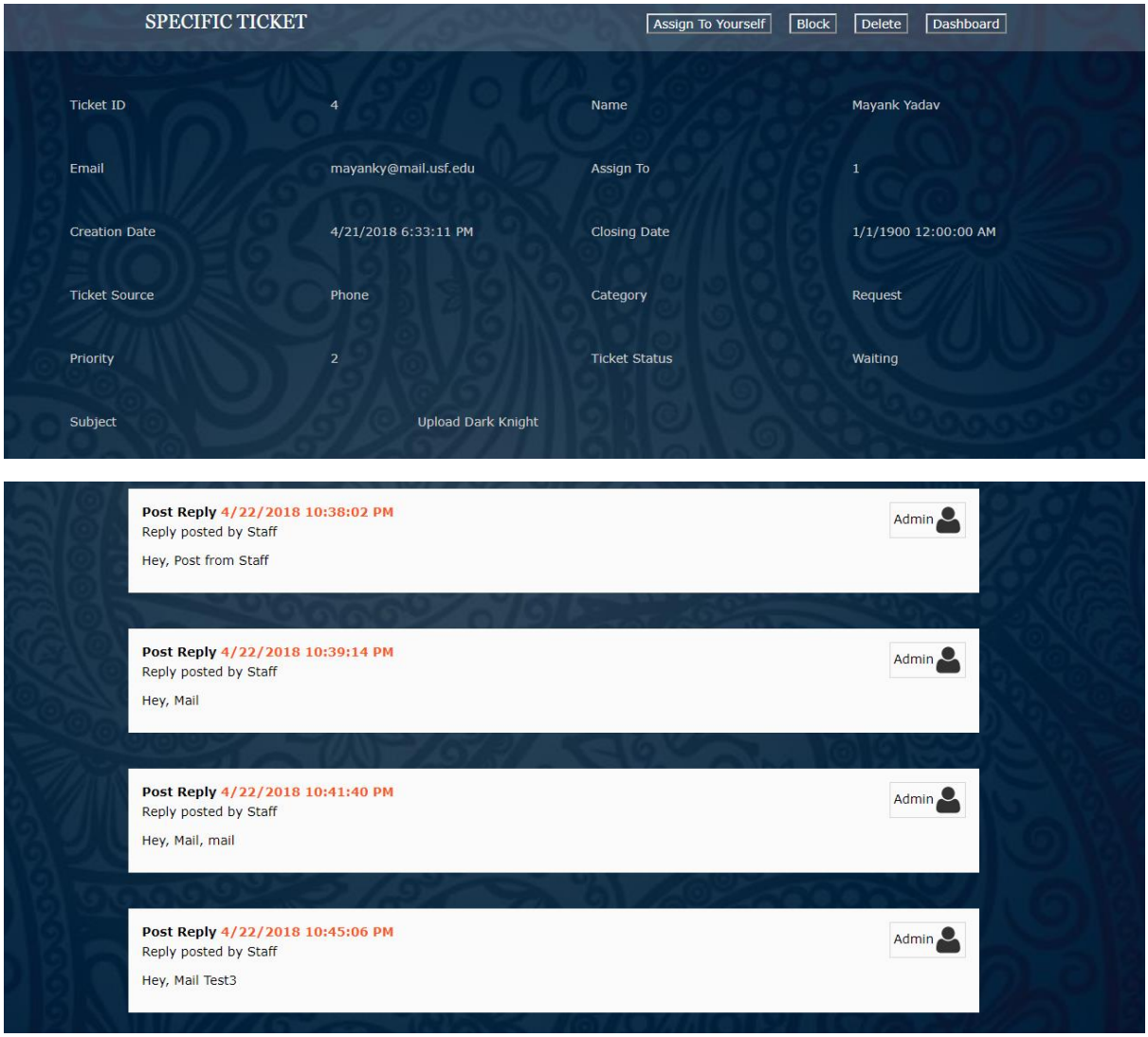


Figure 64 : Specific Ticket

Reply by Staff

- Insert a record in “Ticket_activity” table whenever a staff replies back to a customer’s complaint.

INSERT

INTO

TICKET_Activity

values(

Ticket_Activity_SEQ.NEXTVAL,

'21',

'Reply',

'Reply to Complian about a product',

'It was a company fault. We have talen cre of it.',

Current_Timestamp,

'S', '1', 'N');

Figure 65 : Post reply

Change Ticket Status

INSERT

INTO

TICKET_Activity

values(

Ticket_Activity_SEQ.NEXTVAL,

'21',

'Status',

'Status Changed',

'Your ticket status has been changed to Resolved',

Current_Timestamp,

'S', '1', 'N');

UPDATE TICKET_MASTER SET PRIORITY = '2' WHERE TICKET_ID = '21';

Figure 66 : Ticket Status

Change Ticket Priority

```
INSERT
INTO
TICKET_Activity
values(
Ticket_Activity_SEQ.NEXTVAL,
 '21',
 'Ticket Status',
'Ticket Status has been changed',
'Ticket has been changed to waiting for customer response',Current_Timestamp,
 'S', '1', 'N');


UPDATE TICKET_MASTER SET TICKET_STATUS = 'Waiting' WHERE TICKET_ID =
'21';
```

Figure 67 : Priority

Search in Ticket

- Display only those activities that has the term searched by the user.

```
SELECT
   TICKET_ACTIVITY.ACTIVITY_TYPE,
   TICKET_ACTIVITY.ACTIVITY_DATE,
   (CASE
      WHEN TICKET_ACTIVITY.ACTED_BY = 'C'
      THEN (SELECT
            CLIENT.FIRST_NAME
          FROM
            CLIENT
            INNER JOIN TICKET_MASTER
            ON TICKET_MASTER.CLIENT_ID = CLIENT.CLIENT_ID
         WHERE
            TICKET_MASTER.TICKET_ID = 21)
      WHEN TICKET_ACTIVITY.ACTED_BY = 'S'
      THEN STAFF.USER_NAME
   END) ACTOR,
```

```
   TICKET_ACTIVITY.DESCRIPTION
FROM
   TICKET_ACTIVITY
   LEFT OUTER JOIN STAFF
      ON TICKET_ACTIVITY.ACTOR_ID = STAFF.STAFF_ID
WHERE
   TICKET_ACTIVITY.TICKET_ID = 21 AND (
   TICKET_ACTIVITY.ACTIVITY_TYPE LIKE '%21%' OR
   TICKET_ACTIVITY.ACTIVITY_DATE LIKE '%21%' OR
   STAFF.USER_NAME LIKE '%21%' OR
   TICKET_ACTIVITY.ACTED_BY LIKE '%21%' OR
   TICKET_ACTIVITY.DESCRIPTION LIKE '%21%' OR
   TICKET_ACTIVITY.ACTIVITY_TYPE LIKE UPPER('%21%') OR
   TICKET_ACTIVITY.ACTIVITY_DATE LIKE UPPER('%21%') OR
   STAFF.USER_NAME LIKE UPPER('%21%') OR
   TICKET_ACTIVITY.ACTED_BY LIKE UPPER('%21%') OR
   TICKET_ACTIVITY.DESCRIPTION LIKE UPPER('%21%') OR
   TICKET_ACTIVITY.ACTIVITY_TYPE LIKE LOWER('%21%') OR
   TICKET_ACTIVITY.ACTIVITY_DATE LIKE LOWER('%21%') OR
   STAFF.USER_NAME LIKE LOWER('%21%') OR
   TICKET_ACTIVITY.ACTED_BY LIKE LOWER('%21%') OR
   TICKET_ACTIVITY.DESCRIPTION LIKE LOWER( '%21%'))
ORDER BY
   TICKET_ACTIVITY.ACTIVITY_DATE;
```

Create Ticket

- Create a Ticket via Application Interface

```
INSERT INTO TICKET_MASTER values(
Ticket_Master_SEQ.NEXTVAL,
'43',
```

CURRENT_TIMESTAMP,

'Web',

'Complain','',

'New',

2, '', '',

'N');

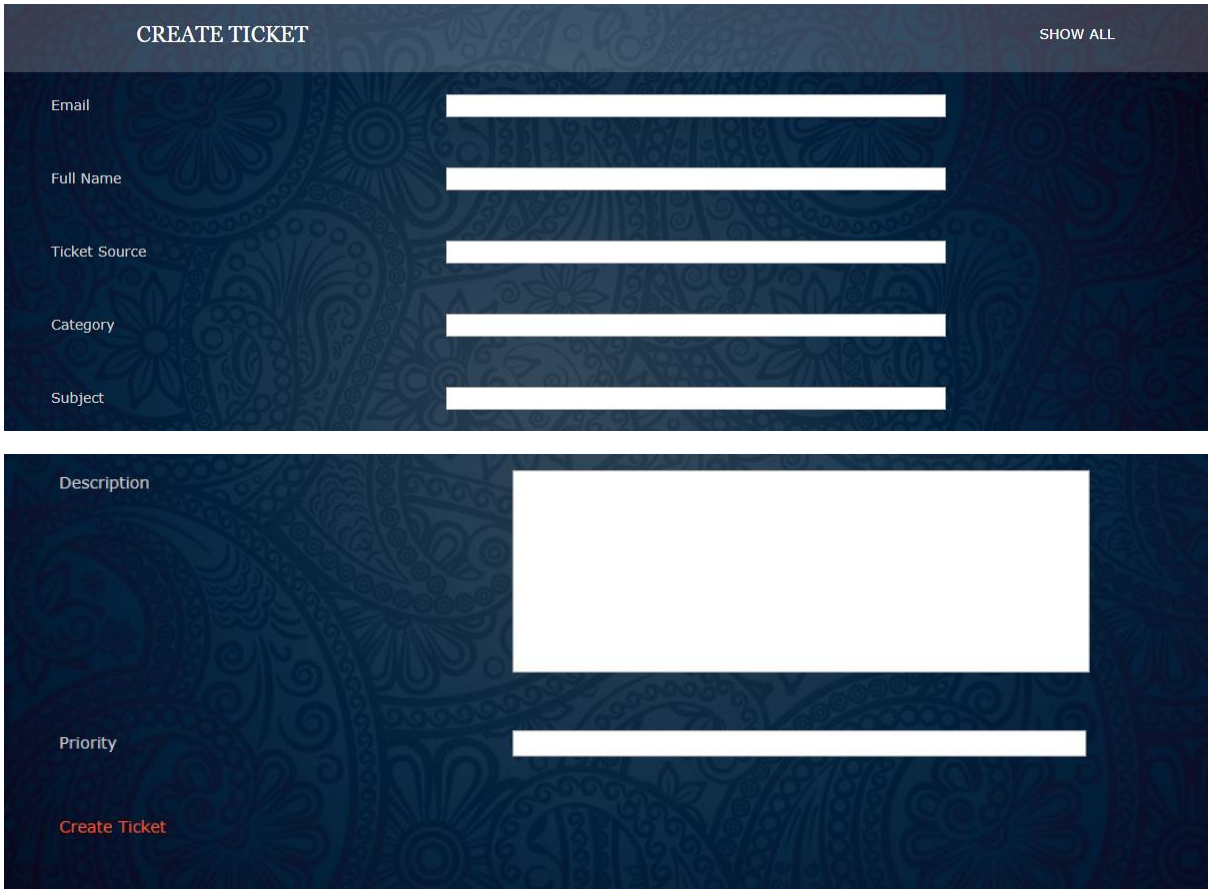


Figure 68 : Create Ticket

- Create a Ticket via Email

INSERT INTO TICKET_MASTER values(

Ticket_Master_SEQ.NEXTVAL,

'43',

CURRENT_TIMESTAMP,

'Email',

'Complain','',

'New',

2, '', '',

'N');

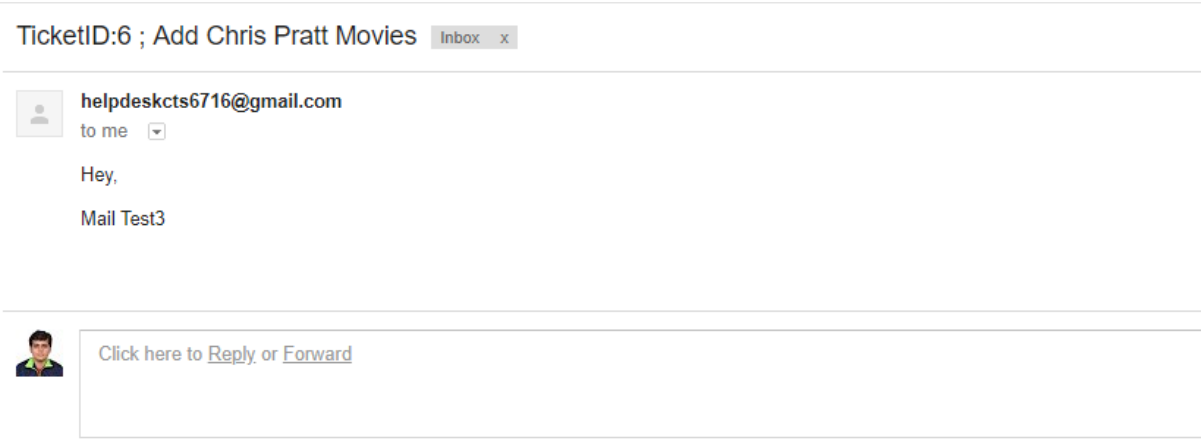


Figure 69 : Email

- Create a Ticket via Phone Call

INSERT INTO TICKET_MASTER values(

Ticket_Master_SEQ.NEXTVAL,

'43',

CURRENT_TIMESTAMP,

'Phone',

'Complain','',

'New',

2, '', '',

'N');

- File an Activity about Ticket Creation

INSERT INTO TICKET_Activity values(

Ticket_Activity_SEQ.NEXTVAL,

'21',

'Create',

'Complian about a product',

'I would like to Complian about a product - coconut hair oil',Current_Timestamp,

'S', 1'', 'N');

Delete a Ticket

- Delete a Specific ticket

```
INSERT INTO TICKET_Activity values(
Ticket_Activity_SEQ.NEXTVAL,
'21',
'Ticket Deleted',
'Ticket has been deleted by the Staff',
'Ticket 21 has been deleted by the Staff 21',
Current_Timestamp, 'S', '1', 'N');
```

- Update the Is_Deleted value to Y for Yes

```
UPDATE TICKET_MASTER
SET IS_DELETED = 'Y'
WHERE TICKET_ID = '21';
```

Block a Client

```
INSERT
 INTO
 Blocklist values(
'21' ,
 '1',
 'POSTING THE TICKETS ON OTHERS COMPANY FORUM',
'N');
```

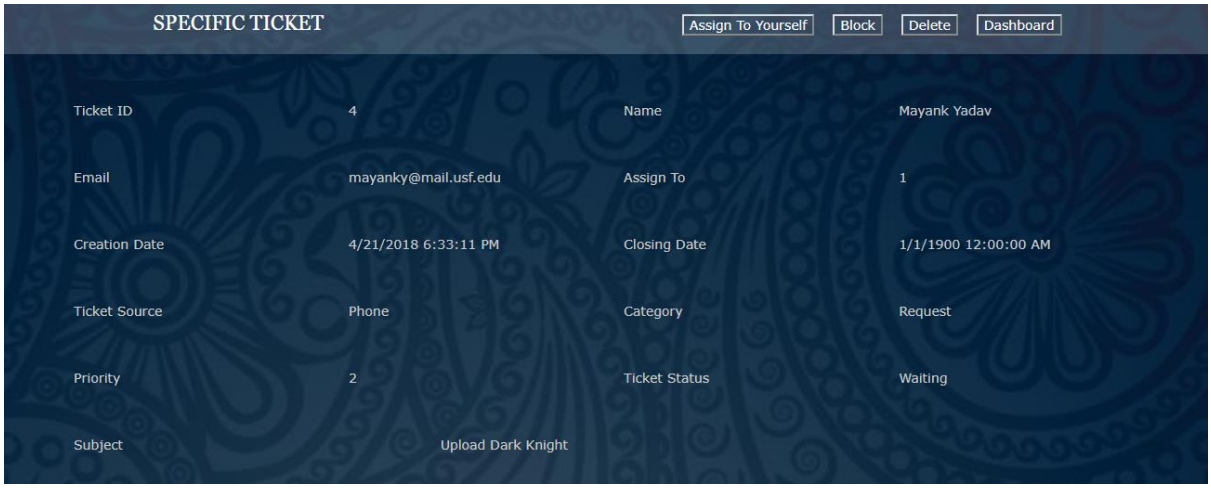


Figure 70 : Delete

Block-list Dashboard

```
SELECT
 CLIENT.FIRST_NAME,
 STAFF.USER_NAME,
 BLOCKLIST.DESCRIPTION
FROM
 BLOCKLIST
 LEFT OUTER JOIN STAFF
  ON STAFF.STAFF_ID = BLOCKLIST.STAFF_ID
 LEFT OUTER JOIN CLIENT
  ON CLIENT.CLIENT_ID = BLOCKLIST.CLIENT_ID
WHERE IS_DELETED = 'N';
```

Figure 71 : Blocklist Dashboard

Remove a Client from Dashboard

```
UPDATE
 BLOCKLIST
 SET
 IS_DELETED = 'N' ,
 STAFF_ID = '1' ,
 DESCRIPTION = 'CUSTOMER CALLED'
WHERE CLIENT_ID = '21';
```

Forgot Password

```
SELECT * FROM DUAL WHERE (SELECT LOWER(SHA(‘New_Password’,256)) FROM DUAL) = (SELECT LOWER(SHA('Confirm_Password',256)) FROM DUAL)

UPDATE
 STAFF
SET
 PASS_WORD = (SELECT LOWER(SHA('new@admin123',256)) FROM DUAL) WHERE
 STAFF_ID = '1' ;
```

Change Password

SELECT * FROM DUAL WHERE (SELECT LOWER(SHA(‘New_Password’,256)) FROM DUAL) = (SELECT LOWER(SHA('Confirm_Password',256)) FROM DUAL)

UPDATE

STAFF

SET PASS_WORD = (SELECT LOWER(SHA('new@admin123’,256)) FROM DUAL) WHERE

PASS_WORD = (SELECT LOWER(SHA(‘admin123’',256)) FROM DUAL) AND STAFF_ID = '1' ;

Show all Clients not in Bblocklist

SELECT FIRST_NAME

FROM CLIENT

WHERE CLIENT.CLIENT_ID NOT IN (SELECT BLOCKLIST.CLIENT_ID FROM BLOCKLIST);

Forum

- ***Client***

Create a Forum

INSERT

INTO Ticket_Forum_MAIN

VALUES

(ticket_forum_main_seq.NEXTVAL,

'43',

'REVIEW OF NIVEA DEO',

'CAN YOU GUYS GIVE ME REVIEW OF THE NIVEA DEO',

CURRENT_TIMESTAMP,

'N');

Figure 72 : Create Forum

Forum Dashboard

```
SELECT
   TICKET_FORUM_MAIN.FORUM_ID,
   CLIENT.FIRST_NAME,
   TICKET_FORUM_MAIN.SUBJECT,
   TICKET_FORUM_MAIN.DESCRIPTION,
   TICKET_FORUM_MAIN.FORUM_DATE
FROM
   TICKET_FORUM_MAIN
      INNER JOIN CLIENT
         ON TICKET_FORUM_MAIN.CLIENT_ID = CLIENT.CLIENT_ID
WHERE
   TICKET_FORUM_MAIN.IS_DELETED = 'N'
ORDER BY
   TICKET_FORUM_MAIN.FORUM_DATE;
```

Figure 73 : Forum Dashboard

Search in Forum Dashboard

```
SELECT
   TICKET_FORUM_MAIN.FORUM_ID,
   CLIENT.FIRST_NAME,
   TICKET_FORUM_MAIN.SUBJECT,
   TICKET_FORUM_MAIN.DESCRIPTION,
   TICKET_FORUM_MAIN.FORUM_DATE
FROM
   TICKET_FORUM_MAIN
      INNER JOIN CLIENT
         ON TICKET_FORUM_MAIN.CLIENT_ID = CLIENT.CLIENT_ID
WHERE
   TICKET_FORUM_MAIN.IS_DELETED = 'N' AND (
   CLIENT.FIRST_NAME LIKE '%TONY%' OR
   TICKET_FORUM_MAIN.SUBJECT LIKE '%TONY%' OR
   TICKET_FORUM_MAIN.DESCRIPTION LIKE '%TONY%' OR
   TICKET_FORUM_MAIN.FORUM_DATE LIKE '%TONY%' OR
   CLIENT.FIRST_NAME LIKE LOWER('%TONY%') OR
```

```
   TICKET_FORUM_MAIN.SUBJECT LIKE LOWER('%TONY%') OR
   TICKET_FORUM_MAIN.DESCRIPTION LIKE LOWER('%TONY%') OR
   TICKET_FORUM_MAIN.FORUM_DATE LIKE LOWER('%TONY%') OR
   CLIENT.FIRST_NAME LIKE UPPER('%TONY%') OR
   TICKET_FORUM_MAIN.SUBJECT LIKE UPPER('%TONY%') OR
   TICKET_FORUM_MAIN.DESCRIPTION LIKE UPPER('%TONY%') OR
   TICKET_FORUM_MAIN.FORUM_DATE LIKE UPPER('%TONY%'))
ORDER BY
   TICKET_FORUM_MAIN.FORUM_DATE;
```

Specific Forum

- Display details related to this Forum

```
SELECT
   TICKET_FORUM_MAIN.FORUM_ID,
   CLIENT.FIRST_NAME,
   TICKET_FORUM_MAIN.SUBJECT,
   TICKET_FORUM_MAIN.DESCRIPTION,
   TICKET_FORUM_MAIN.FORUM_DATE
FROM
   TICKET_FORUM_MAIN
      INNER JOIN CLIENT
         ON TICKET_FORUM_MAIN.CLIENT_ID = CLIENT.CLIENT_ID
WHERE
   TICKET_FORUM_MAIN.FORUM_ID = '3' AND
   TICKET_FORUM_MAIN.IS_DELETED = 'N'
ORDER BY
   TICKET_FORUM_MAIN.FORUM_DATE;
```

- Display all replies related to a particular Forum

```
SELECT
```

```
    TICKET_FORUM_REPLY.REPLY_ID,
    CLIENT.FIRST_NAME,
    TICKET_FORUM_REPLY.DESCRIPTION,
    TICKET_FORUM_REPLY.FORUM_REPLY_DATE
FROM
    TICKET_FORUM_REPLY
     INNER JOIN CLIENT
        ON TICKET_FORUM_REPLY.CLIENT_ID = CLIENT.CLIENT_ID
WHERE
    TICKET_FORUM_REPLY.FORUM_ID = '3' AND
    TICKET_FORUM_REPLY.IS_DELETED = 'N'
ORDER BY
    TICKET_FORUM_REPLY.FORUM_REPLY_DATE;
```

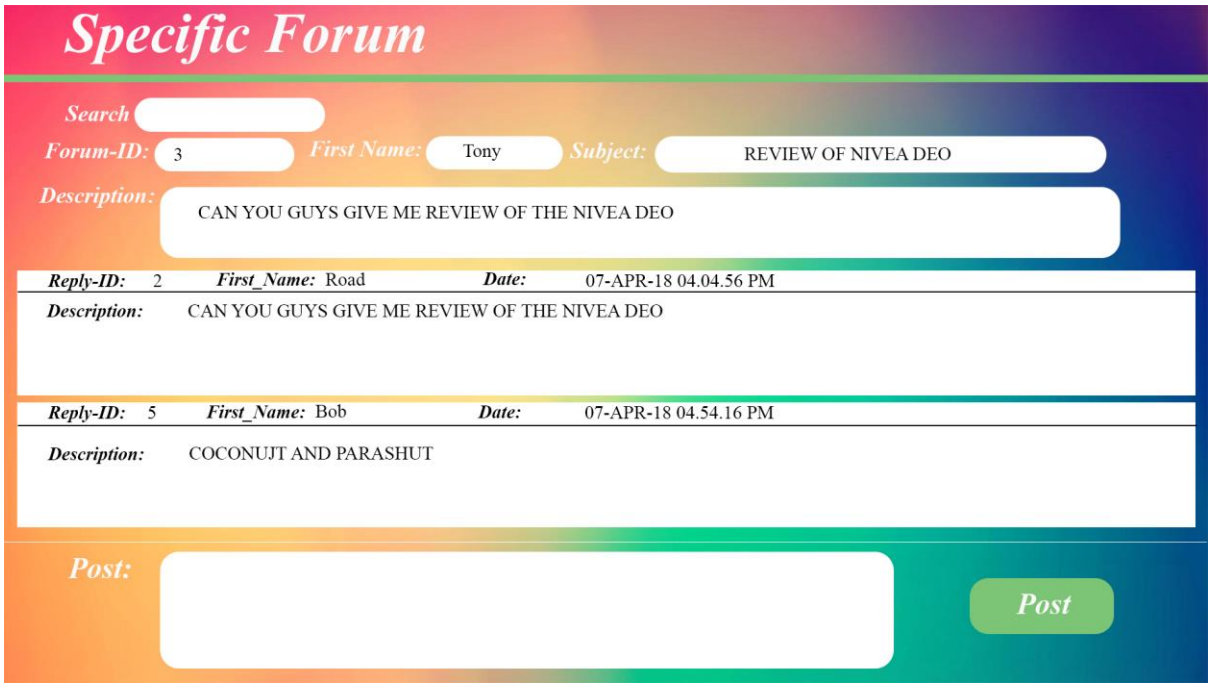


Figure 74 : Specific Forum

Search in Specific Forum

```
SELECT

    TICKET_FORUM_REPLY.REPLY_ID,
```

```
   CLIENT.FIRST_NAME,
   TICKET_FORUM_REPLY.DESCRIPTION,
   TICKET_FORUM_REPLY.FORUM_REPLY_DATE
FROM
   TICKET_FORUM_REPLY
    INNER JOIN CLIENT
      ON TICKET_FORUM_REPLY.CLIENT_ID = CLIENT.CLIENT_ID
WHERE
   TICKET_FORUM_REPLY.FORUM_ID = '3' AND
   TICKET_FORUM_REPLY.IS_DELETED = 'N' AND (
   CLIENT.FIRST_NAME LIKE '%bob%' OR
   TICKET_FORUM_REPLY.DESCRIPTION LIKE '%bob%' OR
   TICKET_FORUM_REPLY.FORUM_REPLY_DATE LIKE '%bob%' OR
   CLIENT.FIRST_NAME LIKE UPPER('%bob%') OR
   TICKET_FORUM_REPLY.DESCRIPTION LIKE UPPER('%bob%') OR
   TICKET_FORUM_REPLY.FORUM_REPLY_DATE LIKE UPPER('%bob%') OR
   CLIENT.FIRST_NAME LIKE LOWER('%bob%') OR
   TICKET_FORUM_REPLY.DESCRIPTION LIKE LOWER('%bob%') OR
   TICKET_FORUM_REPLY.FORUM_REPLY_DATE LIKE LOWER('%bob%'))
ORDER BY
   TICKET_FORUM_REPLY.FORUM_REPLY_DATE;
```

Post on Forum

```
INSERT
 INTO Ticket_Forum_REPLY
VALUES (
ticket_forum_reply_seq.NEXTVAL,
'4',
'45',
'THIS IS GREAT',
```

CURRENT_TIMESTAMP,

'N');

- ***Staff***

Forum Dashboard

```
SELECT
   TICKET_FORUM_MAIN.FORUM_ID,
   CLIENT.FIRST_NAME,
   TICKET_FORUM_MAIN.SUBJECT,
   TICKET_FORUM_MAIN.DESCRIPTION,
   TICKET_FORUM_MAIN.FORUM_DATE
FROM
   TICKET_FORUM_MAIN
      INNER JOIN CLIENT
         ON TICKET_FORUM_MAIN.CLIENT_ID = CLIENT.CLIENT_ID
WHERE
   TICKET_FORUM_MAIN.IS_DELETED = 'N'
ORDER BY
   TICKET_FORUM_MAIN.FORUM_DATE;
```

Figure 75 : Staff Forum Dashboard

Sort Forum

- Sort by Forum ID in ascending order

```
SELECT
   TICKET_FORUM_MAIN.FORUM_ID,
   CLIENT.FIRST_NAME,
   TICKET_FORUM_MAIN.SUBJECT,
   TICKET_FORUM_MAIN.DESCRIPTION,
   TICKET_FORUM_MAIN.FORUM_DATE
FROM
   TICKET_FORUM_MAIN
      INNER JOIN CLIENT
         ON TICKET_FORUM_MAIN.CLIENT_ID = CLIENT.CLIENT_ID
WHERE
   TICKET_FORUM_MAIN.IS_DELETED = 'N'
ORDER BY
   TICKET_FORUM_MAIN.FORUM_ID;
```

- Sort by Forum ID in descending order

```
SELECT
   TICKET_FORUM_MAIN.FORUM_ID,
   CLIENT.FIRST_NAME,
   TICKET_FORUM_MAIN.SUBJECT,
   TICKET_FORUM_MAIN.DESCRIPTION,
   TICKET_FORUM_MAIN.FORUM_DATE
FROM
   TICKET_FORUM_MAIN
      INNER JOIN CLIENT
         ON TICKET_FORUM_MAIN.CLIENT_ID = CLIENT.CLIENT_ID
WHERE
   TICKET_FORUM_MAIN.IS_DELETED = 'N'
ORDER BY
   TICKET_FORUM_MAIN.FORUM_ID DESC;
```

- Sort by Client's First Name in ascending order

```
SELECT
   TICKET_FORUM_MAIN.FORUM_ID,
   CLIENT.FIRST_NAME,
   TICKET_FORUM_MAIN.SUBJECT,
   TICKET_FORUM_MAIN.DESCRIPTION,
   TICKET_FORUM_MAIN.FORUM_DATE
FROM
   TICKET_FORUM_MAIN
      INNER JOIN CLIENT
         ON TICKET_FORUM_MAIN.CLIENT_ID = CLIENT.CLIENT_ID
WHERE
   TICKET_FORUM_MAIN.IS_DELETED = 'N'
ORDER BY
   CLIENT.FIRST_NAME;
```

- Sort by Client’s First Name in descending order

```
SELECT
   TICKET_FORUM_MAIN.FORUM_ID,
   CLIENT.FIRST_NAME,
   TICKET_FORUM_MAIN.SUBJECT,
   TICKET_FORUM_MAIN.DESCRIPTION,
   TICKET_FORUM_MAIN.FORUM_DATE
FROM
   TICKET_FORUM_MAIN
      INNER JOIN CLIENT
         ON TICKET_FORUM_MAIN.CLIENT_ID = CLIENT.CLIENT_ID
WHERE
   TICKET_FORUM_MAIN.IS_DELETED = 'N'
ORDER BY
   CLIENT.FIRST_NAME DESC;
```

- Sort by Subject in ascending order

```
SELECT
   TICKET_FORUM_MAIN.FORUM_ID,
   CLIENT.FIRST_NAME,
   TICKET_FORUM_MAIN.SUBJECT,
   TICKET_FORUM_MAIN.DESCRIPTION,
   TICKET_FORUM_MAIN.FORUM_DATE
FROM
   TICKET_FORUM_MAIN
      INNER JOIN CLIENT
         ON TICKET_FORUM_MAIN.CLIENT_ID = CLIENT.CLIENT_ID
WHERE
   TICKET_FORUM_MAIN.IS_DELETED = 'N'
```

```
ORDER BY
   TICKET_FORUM_MAIN.SUBJECT;
```

- Sort by Subject in descending order

```
SELECT
   TICKET_FORUM_MAIN.FORUM_ID,
   CLIENT.FIRST_NAME,
   TICKET_FORUM_MAIN.SUBJECT,
   TICKET_FORUM_MAIN.DESCRIPTION,
   TICKET_FORUM_MAIN.FORUM_DATE
FROM
   TICKET_FORUM_MAIN
      INNER JOIN CLIENT
         ON TICKET_FORUM_MAIN.CLIENT_ID = CLIENT.CLIENT_ID
WHERE
   TICKET_FORUM_MAIN.IS_DELETED = 'N'
ORDER BY
   TICKET_FORUM_MAIN.SUBJECT DESC;
```

- Sort by Description in ascending order

```
SELECT
   TICKET_FORUM_MAIN.FORUM_ID,
   CLIENT.FIRST_NAME,
   TICKET_FORUM_MAIN.SUBJECT,
   TICKET_FORUM_MAIN.DESCRIPTION,
   TICKET_FORUM_MAIN.FORUM_DATE
FROM
   TICKET_FORUM_MAIN
      INNER JOIN CLIENT
         ON TICKET_FORUM_MAIN.CLIENT_ID = CLIENT.CLIENT_ID
```

```
WHERE
   TICKET_FORUM_MAIN.IS_DELETED = 'N'
ORDER BY
   TICKET_FORUM_MAIN.DESCRIPTION;
```

- Sort by Description in descending order

```
SELECT
   TICKET_FORUM_MAIN.FORUM_ID,
   CLIENT.FIRST_NAME,
   TICKET_FORUM_MAIN.SUBJECT,
   TICKET_FORUM_MAIN.DESCRIPTION,
   TICKET_FORUM_MAIN.FORUM_DATE
FROM
   TICKET_FORUM_MAIN
      INNER JOIN CLIENT
         ON TICKET_FORUM_MAIN.CLIENT_ID = CLIENT.CLIENT_ID
WHERE
   TICKET_FORUM_MAIN.IS_DELETED = 'N'
ORDER BY
   TICKET_FORUM_MAIN.DESCRIPTION DESC;
```

- Sort by Forum Date in ascending order

```
SELECT
   TICKET_FORUM_MAIN.FORUM_ID,
   CLIENT.FIRST_NAME,
   TICKET_FORUM_MAIN.SUBJECT,
   TICKET_FORUM_MAIN.DESCRIPTION,
   TICKET_FORUM_MAIN.FORUM_DATE
FROM
   TICKET_FORUM_MAIN
```

```
    INNER JOIN CLIENT
        ON TICKET_FORUM_MAIN.CLIENT_ID = CLIENT.CLIENT_ID
WHERE
  TICKET_FORUM_MAIN.IS_DELETED = 'N'
ORDER BY
  TICKET_FORUM_MAIN.FORUM_DATE ASC;
```

- Sort by Forum Date in descending order

```
SELECT
  TICKET_FORUM_MAIN.FORUM_ID,
  CLIENT.FIRST_NAME,
  TICKET_FORUM_MAIN.SUBJECT,
  TICKET_FORUM_MAIN.DESCRIPTION,
  TICKET_FORUM_MAIN.FORUM_DATE
FROM
  TICKET_FORUM_MAIN
    INNER JOIN CLIENT
      ON TICKET_FORUM_MAIN.CLIENT_ID = CLIENT.CLIENT_ID
WHERE
  TICKET_FORUM_MAIN.IS_DELETED = 'N'
ORDER BY
  TICKET_FORUM_MAIN.FORUM_DATE DESC;
```

Search in Forum Dashboard

```
SELECT
  TICKET_FORUM_MAIN.FORUM_ID,
  CLIENT.FIRST_NAME,
  TICKET_FORUM_MAIN.SUBJECT,
  TICKET_FORUM_MAIN.DESCRIPTION,
  TICKET_FORUM_MAIN.FORUM_DATE
```

```
FROM
   TICKET_FORUM_MAIN
      INNER JOIN CLIENT
         ON TICKET_FORUM_MAIN.CLIENT_ID = CLIENT.CLIENT_ID
WHERE
   TICKET_FORUM_MAIN.IS_DELETED = 'N' AND (
   CLIENT.FIRST_NAME LIKE '%TONY%' OR
   TICKET_FORUM_MAIN.SUBJECT LIKE '%TONY%' OR
   TICKET_FORUM_MAIN.DESCRIPTION LIKE '%TONY%' OR
   TICKET_FORUM_MAIN.FORUM_DATE LIKE '%TONY%' OR
   CLIENT.FIRST_NAME LIKE LOWER('%TONY%') OR
   TICKET_FORUM_MAIN.SUBJECT LIKE LOWER('%TONY%') OR
   TICKET_FORUM_MAIN.DESCRIPTION LIKE LOWER('%TONY%') OR
   TICKET_FORUM_MAIN.FORUM_DATE LIKE LOWER('%TONY%') OR
   CLIENT.FIRST_NAME LIKE UPPER('%TONY%') OR
   TICKET_FORUM_MAIN.SUBJECT LIKE UPPER('%TONY%') OR
   TICKET_FORUM_MAIN.DESCRIPTION LIKE UPPER('%TONY%') OR
   TICKET_FORUM_MAIN.FORUM_DATE LIKE UPPER('%TONY%'))
ORDER BY
   TICKET_FORUM_MAIN.FORUM_DATE;
```

Specific Forum

- Display all the details related to a particular Forum

```
SELECT
   TICKET_FORUM_MAIN.FORUM_ID,
   CLIENT.FIRST_NAME,
   TICKET_FORUM_MAIN.SUBJECT,
   TICKET_FORUM_MAIN.DESCRIPTION,
   TICKET_FORUM_MAIN.FORUM_DATE
FROM
```

```
   TICKET_FORUM_MAIN
      INNER JOIN CLIENT
         ON TICKET_FORUM_MAIN.CLIENT_ID = CLIENT.CLIENT_ID
WHERE
   TICKET_FORUM_MAIN.FORUM_ID = '3' AND
   TICKET_FORUM_MAIN.IS_DELETED = 'N'
ORDER BY
   TICKET_FORUM_MAIN.FORUM_DATE;
```

- Display all reply related to a particular Forum

```
SELECT

   TICKET_FORUM_REPLY.REPLY_ID,
   CLIENT.FIRST_NAME,
   TICKET_FORUM_REPLY.DESCRIPTION,
   TICKET_FORUM_REPLY.FORUM_REPLY_DATE
FROM
   TICKET_FORUM_REPLY
    INNER JOIN CLIENT
      ON TICKET_FORUM_REPLY.CLIENT_ID = CLIENT.CLIENT_ID
WHERE
   TICKET_FORUM_REPLY.FORUM_ID = '3' AND
   TICKET_FORUM_REPLY.IS_DELETED = 'N'
ORDER BY
   TICKET_FORUM_REPLY.FORUM_REPLY_DATE;
```

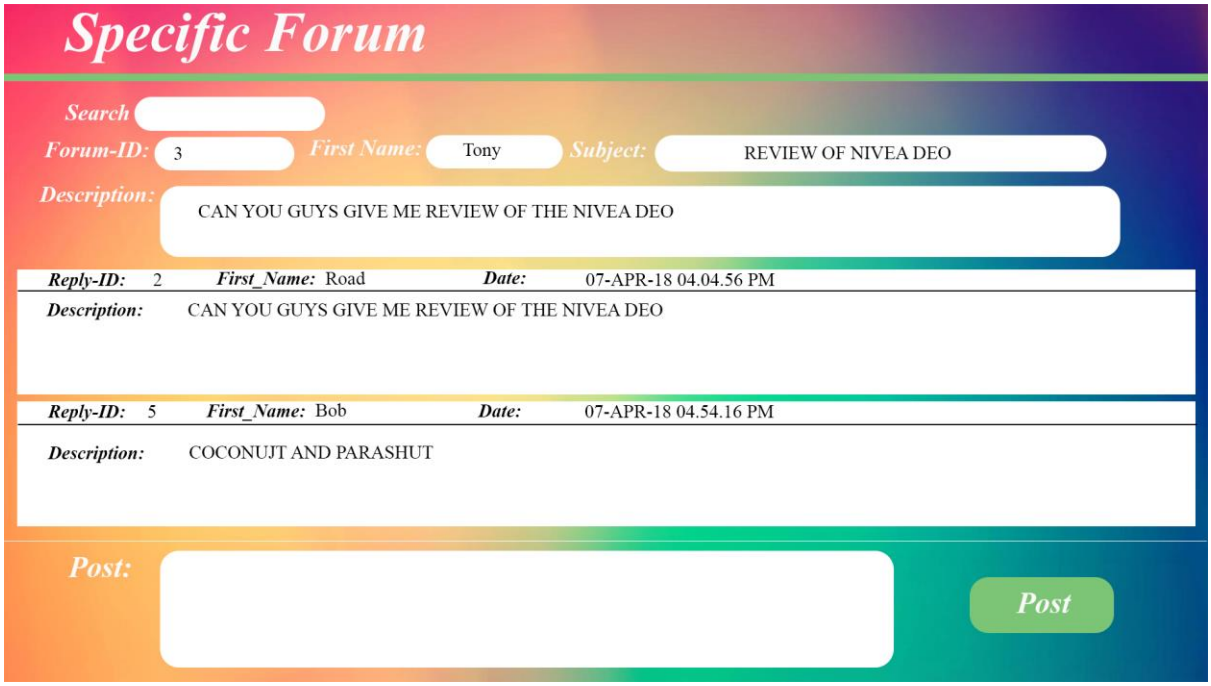


Figure 76 : Staff Specific Forum

Search in Specific Forum

```
SELECT

   TICKET_FORUM_REPLY.REPLY_ID,
   CLIENT.FIRST_NAME,
   TICKET_FORUM_REPLY.DESCRIPTION,
   TICKET_FORUM_REPLY.FORUM_REPLY_DATE
FROM
   TICKET_FORUM_REPLY
    INNER JOIN CLIENT
      ON TICKET_FORUM_REPLY.CLIENT_ID = CLIENT.CLIENT_ID
WHERE
   TICKET_FORUM_REPLY.FORUM_ID = '3' AND
   TICKET_FORUM_REPLY.IS_DELETED = 'N' AND (
   CLIENT.FIRST_NAME LIKE '%bob%' OR
   TICKET_FORUM_REPLY.DESCRIPTION LIKE '%bob%' OR
   TICKET_FORUM_REPLY.FORUM_REPLY_DATE LIKE '%bob%' OR
   CLIENT.FIRST_NAME LIKE UPPER('%bob%') OR
```

```
   TICKET_FORUM_REPLY.DESCRIPTION LIKE UPPER('%bob%') OR
   TICKET_FORUM_REPLY.FORUM_REPLY_DATE LIKE UPPER('%bob%') OR
   CLIENT.FIRST_NAME LIKE LOWER('%bob%') OR
   TICKET_FORUM_REPLY.DESCRIPTION LIKE LOWER('%bob%') OR
   TICKET_FORUM_REPLY.FORUM_REPLY_DATE LIKE LOWER('%bob%'))
ORDER BY
   TICKET_FORUM_REPLY.FORUM_REPLY_DATE;
```

Delete a Forum

```
UPDATE
 TICKET_FORUM_MAIN
SET IS_DELETED = 'Y'
WHERE FORUM_ID = '3';
```

Delete a Reply

```
UPDATE
 TICKET_FORUM_REPLY
SET IS_DELETED = 'Y'
WHERE REPLY_ID = '2';
```

DATABASE PROGRAMMING

1. Trigger – Check Client Birthday.
   Client date of birth must be later than Jan 1, 1900 and earlier than today

```
CREATE OR REPLACE TRIGGER check_client_birth_date
 BEFORE INSERT OR UPDATE ON client
 FOR EACH ROW
BEGIN
 IF( :new.birth_date < date '1900-01-01' or
    :new.birth_date > sysdate )
 THEN
  RAISE_APPLICATION_ERROR(
    -20001,
    'Client date of birth must be later than Jan 1, 1900 and earlier than today' );
 END IF;
END;
```

2. Function - Store Password in hash form

```
CREATE OR REPLACE FUNCTION "SHA" (p_string IN VARCHAR2, p_bits IN
NUMBER)
  RETURN RAW
AS
  LANGUAGE JAVA
  NAME 'sha.get_digest( java.lang.String, int ) return oracle.sql.RAW' ;

CREATE OR REPLACE AND COMPILE  JAVA SOURCE NAMED sha
  AS import java.security.MessageDigest;
import oracle.sql.*;

public class sha
{
```

```
public static oracle.sql.RAW get_digest( String p_string, int p_bits ) throws Exception
{
MessageDigest v_md = MessageDigest.getInstance( "SHA-" + p_bits );
byte[] v_digest;
v_digest = v_md.digest( p_string.getBytes( "UTF-8" ) );
return RAW.newRAW(v_digest);
}
}
/
```

--In 256 Encryption will get 64 Character Length String

```
SQL> SELECT LOWER(SHA('Oracle PL/SQL World',256)) FROM DUAL;
1f2f1fd5ab78cf15bd3cdf330d6ab79ba60cc797fb4875e5267f90daa10157aa
```

--In 512 Encryption will get 128 Character Length String

```
SQL> SELECT LOWER(SHA('Oracle PL/SQL World',512)) FROM DUAL;
478811ecf93ae1564ad0775c20aa0f1b5de5992306974a9339b25794de32a000d058136c48122
9c38f3ee9e6775426a3b54ff1daf495d3b7f2b6e571595ca888
```

3. Procedure – Get alert for Client's birthdate

```
CREATE OR REPLACE PROCEDURE myproc AS
BEGIN
SELECT *
FROM client
WHERE to_char(birth_date,'dd-mon') = (select to_char(sysdate,'dd-mon') from dual);
END myproc;
```

4. Sequences
   - Sequence for Staff
     ```
     CREATE SEQUENCE staff_seq
       START WITH 1
       INCREMENT BY 1;
     ```
   - Sequence for Client
     ```
     CREATE SEQUENCE client_seq
       START WITH 1
     ```

```
INCREMENT BY 1;
```

- Sequence for Block list

```
CREATE SEQUENCE blocklist_seq
 START WITH 1
 INCREMENT BY 1;
```

- Sequence for Ticket Main

```
CREATE SEQUENCE ticket_main_seq
 START WITH 1
 INCREMENT BY 1;
```

- Sequence for Ticket Activity

```
CREATE SEQUENCE ticket_activity_seq
 START WITH 1
 INCREMENT BY 1;
```

- Sequence for Ticket Forum Main

```
CREATE SEQUENCE ticket_forum_main_seq
 START WITH 1
 INCREMENT BY 1;
```

- Sequence for Ticket Forum Reply

```
CREATE SEQUENCE ticket_forum_reply_seq
 START WITH 1
 INCREMENT BY 1;
```

5. Create New Ticket. Can be from Client Side and Staff Side

```
CREATE OR REPLACE PROCEDURE InsertNewTicket(
CLIENT_ID NUMBER, CATEGORIES VARCHAR2, TICKET_SOURCE
VARCHAR2, PRIORITY NUMBER,
SUBJECT VARCHAR2, DESCRIPTION VARCHAR2, ACTED_BY CHAR,
ACTOR_ID NUMBER)
IS
BEGIN
INSERT INTO TICKET_MASTER
values(Ticket_Master_SEQ.NEXTVAL, CLIENT_ID,
CURRENT_TIMESTAMP, TICKET_SOURCE,
CATEGORIES,'','New',PRIORITY, '', '','N');
INSERT INTO TICKET_Activity
values(Ticket_Activity_SEQ.NEXTVAL, Ticket_Master_SEQ.CURRVAL,
'Create', Subject, DESCRIPTION, CURRENT_TIMESTAMP,
ACTED_BY, ACTOR_ID, 'N');
commit;
END;
/
```

6. Block a Client from Creating New Tickets

```
CREATE OR REPLACE PROCEDURE BlockCleint(
CLIENT_ID NUMBER, STAFF_ID NUMBER, DESCRIPTION VARCHAR2)
IS
BEGIN
INSERT
 INTO
 Blocklist values(
 CLIENT_ID,
 STAFF_ID,
 DESCRIPTION,
'N');
commit;
END;
/
```

7. Create a New Forum

```
CREATE OR REPLACE PROCEDURE CreateNewForum(
CLIENT_ID NUMBER, SUBJECT VARCHAR2, DESCRIPTION VARCHAR2)
IS
BEGIN
INSERT
 INTO Ticket_Forum_MAIN
VALUES
(ticket_forum_main_seq.NEXTVAL,
CLIENT_ID,
SUBJECT,
DESCRIPTION,
CURRENT_TIMESTAMP,
'N');
commit;
END;
/
```

8. Create New Forum Reply

```
CREATE OR REPLACE PROCEDURE CreateNewForumReply(
FORUM_ID NUMBER, CLIENT_ID NUMBER, DESCRIPTION VARCHAR2)
IS
BEGIN
INSERT
 INTO Ticket_Forum_REPLY
VALUES (
ticket_forum_reply_seq.NEXTVAL,
FORUM_ID,
CLIENT_ID,
DESCRIPTION,
```

```
CURRENT_TIMESTAMP,
'N');
commit;
END;
/
```

9. Delete a Ticket

```
CREATE OR REPLACE PROCEDURE DeleteTicket(
TICKET_ID NUMBER, Staff_ID NUMBER, SUBJECT VARCHAR2,
DESCRIPTION VARCHAR2, ACTED_BY CHAR, ACTOR_ID NUMBER
)
IS
begin
INSERT INTO TICKET_Activity values(
Ticket_Activity_SEQ.NEXTVAL,
TICKET_ID,
'Ticket Deleted',
SUBJECT,
DESCRIPTION,
Current_Timestamp, ACTED_BY, ACTOR_ID, 'N');
UPDATE TICKET_MASTER
SET IS_DELETED = 'Y'
WHERE TICKET_ID = TICKET_ID;
commit;
END;
/
```

10. Delete a Forum

```
CREATE OR REPLACE PROCEDURE DeleteForum(
FORUM_ID NUMBER)
IS
begin
UPDATE
 TICKET_FORUM_MAIN
SET IS_DELETED = 'Y'
WHERE FORUM_ID = FORUM_ID;
commit;
END;
/
```

11. Delete a Reply

```
CREATE OR REPLACE PROCEDURE DeleteForumReply(
REPLY_ID NUMBER)
IS
begin
UPDATE
 TICKET_FORUM_REPLY
SET IS_DELETED = 'Y'
WHERE REPLY_ID = REPLY_ID;
commit;
END;
/
```

INTERESTING QUERIES

1. Show all Tickets (including the ones that been deleted) and activities performed on them where client is blocked.

```
SELECT
 *
FROM
 TICKET_ACTIVITY
WHERE
 TICKET_ID =
(SELECT
 TICKET_MASTER.TICKET_ID
FROM
 TICKET_MASTER
INNER JOIN BLOCKLIST
 ON TICKET_MASTER.CLIENT_ID = BLOCKLIST.CLIENT_ID
);
```

2. Delete all tickets of a blocked client

```
SELECT
 TICKET_MASTER.TICKET_ID
FROM TICKET_MASTER
INNER JOIN BLOCKLIST
 ON TICKET_MASTER.CLIENT_ID = BLOCKLIST.CLIENT_ID
UPDATE TICKET_MASTER SET IS_DELETED = 'Y';
```

3. Show Forum and Reply created by a blocked client

```
SELECT CLIENT_ID, DESCRIPTION, FORUM_DATE
FROM TICKET_FORUM_MAIN
WHERE CLIENT_ID
IN (
SELECT TICKET_FORUM_MAIN.CLIENT_ID
FROM TICKET_FORUM_MAIN
INNER JOIN BLOCKLIST
 ON TICKET_FORUM_MAIN.CLIENT_ID = BLOCKLIST.CLIENT_ID);

SELECT CLIENT_ID, DESCRIPTION, FORUM_DATE
FROM TICKET_FORUM_REPLY
WHERE CLIENT_ID
IN (
SELECT TICKET_FORUM_REPLY.CLIENT_ID
FROM TICKET_FORUM_REPLY
```

INNER JOIN BLOCKLIST
ON TICKET_FORUM_REPLY.CLIENT_ID = BLOCKLIST.CLIENT_ID);

4. Delete all Forum and Reply if a Client is Blocked

UPDATE TICKET_FORUM_MAIN SET IS_DELETED = 'Y' WHERE CLIENT_ID IN (

SELECT TICKET_FORUM_MAIN.CLIENT_ID FROM TICKET_FORUM_MAIN

INNER JOIN BLOCKLIST

ON TICKET_FORUM_MAIN.CLIENT_ID = BLOCKLIST.CLIENT_ID);

UPDATE TICKET_FORUM_REPLY SET IS_DELETED = 'Y' WHERE CLIENT_ID IN (

SELECT TICKET_FORUM_REPLY.CLIENT_ID FROM TICKET_FORUM_REPLY

INNER JOIN BLOCKLIST

ON TICKET_FORUM_REPLY.CLIENT_ID = BLOCKLIST.CLIENT_ID);

ANNEXURE

FileStream

The FILESTREAM data type is Microsoft's answer to SQL Server BLOB storage. The FILESTREAM data type lets you combine the best of both worlds. It uses the higher performance native NTFS file system to store and access BLOB data which means that the size limitations are lifted to maximum file size supported by the NTFS file systems which is 16 TB. It removes the need to process BLOB data in the Buffer Pool. At the same time it provides you with full data consistency. The FILESTREAM data type stores pointers in each column that connect the relational data in the row to the FILESTREAM data stored in the file system. This enables that data relationships to be consistent and durable. For instance, if you backup a database that's using the FILESTREAM data type all of the file system data will be backed up right along with your relational data [4].

Consideration for using the FILESTREAM Data Type

Like you might expect there are some considerations for using FILESTREAM data.

- Table with FILESTREAM columns must have a nonnull unique row ID
- FILESTREAM filegroups support multiple data containers
- FILESTREAM data containers cannot be nested
- FILESTREAM filegroups are supported with failover clustering but the filegroups must be on shared disk resources
- FILESTREAM filegroups are supported by AlwaysOn Availability Groups
- FILESTREAM filegroups can be on compressed volumes
- FILESTREAM filegroups and containers should reside on volumes other than the operating system, paging file, SQL Server database, SQL Server log, or tempdb

REFERENCES

1. https://github.com/arulxaviers/Oracle_PLSQL_SHA_256_512_Encryption
2. https://www.w3schools.com/sql/sql_unique.asp
3. https://www.youtube.com/watch?v=Nwmt5S4Mod8
4. https://www.experts-exchange.com/questions/24642557/SQL-SYSDATE-to-find-out-birthdays-in-the-next-30days.html
5. https://docs.microsoft.com/en-us/sql/relational-databases/blob/filestream-sql-server?view=sql-server-2017
6. https://mockaroo.com/

PROJECT ASSESSMENT

Topic / Section	Description	Evaluation
Logical database design	The logical design section should include entity-relationship diagrams (ERDs) and data dictionaries for your database design, as well as any design assumptions. There should also be a complete ERD for your entire project. There is no expectation that you implement all of your design, just indicate the areas built.	20
Physical database design	This section should cover implementation-level issues. For instance, discuss predicted usage and indexing strategies that support expected activities. In addition, you may wish to discuss architecture issues, including distributed database issues (even though you may not implement anything in these areas). Artifacts could include capacity planning, storage subsystems, and data placement (e.g., tablespace / file system arrangements), indexing strategies, transaction usage maps, etc.	10
Data generation and loading	Describe the queries, stored procedures, desktop tools (e.g., MS Excel) that were used to populate the database. You may have used queries with mod function, data arithmetic, number sequences, lookup tables, and even data from the Web. Any / all of these are interesting additions to the project. You must create and populate at least five tables from your design. Two of those tables must include at least 10,000 records a piece. Include a count of the number of rows inserted into each table.	15
Performance tuning	In this section, highlight any experiments run as part of the project related to performance tuning. Experiments with different indexing strategies, optimizer changes, transaction isolation levels, function-based indexes, and table partitioning can all be interesting. Remember to look at different types of queries (e.g., point, range, scan), execution plans, and I/O burden. For each experiment include the following: (1) purpose of the experiment, (2) steps followed to run the experiment, (3) key results (include screenshots, figures, and/or tables to help highlight results), and (4) a discussion of the results that explains what happened and why.	15
Querying	In this section, create queries that highlight the types of questions that can be answered by the database. These queries should demonstrate your skills in query writing. (Analytic SQL extensions may be explored for this section.)	20
DBA scripts	During the semester, we looked at example DBA scripts that query the system catalog (a good way to explore the database engine). Provide DBA scripts that are helpful for reporting on database objects, indexes, constraints, physical storage, data	10

	files, etc. For each script provide the following: (1) SQL / PL/SQL code, (2) description of why the script is useful, (3) how the script could be used, and (4) some sample results from executing the script.	
Database programming	For this section, highlight any stored procedures, functions, or triggers that were created that are not included in the data generation and loading topic.	15
Database security	Database security is an important area of interest that can also be investigated. Though you are limited on the implementation side, you can develop a security policy and discuss how you would implement various aspects using authentication strategies, roles, profiles, and even auditing features.	5
Interface design / Data visualization	Though interface issues are not typically the focus of the project, you are free to add emphasis here. You can do everything from sketches and mock-ups, to using HTML and other web-enabled tools to build an interface. You can also experiment with creating visualizations for your data using a variety of freely-available tools such as Tableau Public.	15