INDRAPRASTHA INSTITUTE *of*
INFORMATION TECHNOLOGY **DELHI**

PROJECT REPORT

DATABASE MANAGEMENT SYSTEM

TEAM NACHOS PRESENTS

INSURE-X

A HOLISTIC INSURANCE FIRM

IN ASSOCIATION WITH

DEPARTMENT OF COMPUTER SCIENCE, IIITD

CO-AUTHORS

KHUSHALI VERMA | 2018290

AYAAN KAKKAR | 2018028

PRAGYAN MEHROTRA | 2018168

RITIK KHANNA | 2018084

AAKASH KUMAR SAGAR | 2018323

DECLARATION

We hereby declare that the project titled "INSURE-X : A HOLISTIC INSURANCE FIRM" is the result of the collective efforts of the entire team, under the guidance of Dr. Md. Shad Akhtar and Dr. Mukesh Mohania, our instructors. We also assure that the outcomes are a result of our own research and analysis conducted throughout the course of the project.

Team Nachos
(Group-20)

ACKNOWLEDGEMENT

First and foremost, we would like to express our sincere gratitude to both the instructors - Dr. Md. Shad Akhtar and Dr. Mukesh Mohania, for consistently guiding us through the project and giving us the right direction. Next, we would also like to thank the teaching assistants for helping us overcome various hurdles throughout the project.

Kudos to the entire team for cooperating throughout the project and showing an excellent team spirit. It would have been impossible without the continuous synergy of all!

Last but not the least, we would like to thank all those people, who spared time out of their schedule to help us at one point of time or the other, in conducting various studies and/or solving issues persistently. You all are of great value to this project!

Thank you all once again!

ABSTRACT

The agenda behind this project is to develop a robust insurance management system, as a web based application, which has a more user-friendly interface, and is easy to navigate for the end-users, as well as easy to manage for the administrators and officials.

To begin with the project, a proper design process has been followed to come up with the various-level stakeholders and issues that exist in the current insurance management systems. Throughout the course of the project, efforts have been put in the best of our efforts in resolving these issues, and provide a better experience to each and every stakeholder while they use the system.

ROLE DISTRIBUTION FOR THE PROJECT

- Ayaan Kakkar : Back-end and Front-end
- Pragyan Mehrotra : SQL Queries and Data Population
- Khushali Verma : Queries and W-in-P Document
- Ritik Khanna : Front-end
- Aakash Kumar Sagar : Front-end

TABLE OF CONTENTS

WEEK 1

DECIDING THE PROJECT DOMAIN

Database Management Systems are deeply embedded into every real-world model - may it be a library at a college, or apps like Uber and Zomato, databases are the fundamentals of their systems. Without maintaining and managing databases, neither would Banks be able to function , nor would you be able to stream your favorite music or videos online. Among all of these and many more, it was important to choose a database that is integral to a large number of audience, and has a scope for improvement through this project.

For this purpose, apart from the research we did over the internet, we followed these design processes to come up with the best suitable domain -

CROWDSTORMING

Crowdstorming session involves the audience to critique and comment (or give feedback) over some product (both digital and hardware). User feedback is important at every stage, and conducting a crowdstorming session basically brings different kinds of personas together to discuss and debate over certain things that they may be perceiving differently. Different people, from varied backgrounds and age groups were considered at this stage and a few of the domains that we identified after the crowdstorming session were -

- Job/Internship Portals
- ERP (specific to IIITD)
- Warehouse Management
- Insurance Management
- Pharmacy Management
- Library Management
- Online Course Offering Management

NOISE ANALYSIS

NOISE Analysis is a way to measure certain aspects of a product or a scenario. NOISE here means Needs, Opportunities, Improvement, Strengths and Exceptions. This process gives a holistic overview of the situation - what is the requirements, what is it that can or cannot be done, what can be improved and what are its strengths. We carried out NOISE Analysis for all the listed domains, and the domain which had the most diverse audience coupled with a lot of upcoming needs and opportunities was -

INSURANCE MANAGEMENT SYSTEM

and named our system as

"INSURE-X"

PROBLEM STATEMENT & OBJECTIVE

The current insurance management system has various persistent issues as identified after user interviews. Various user-groups who are somewhere connected to the insurance system were either interviewed face to face or over a phone call, to get insights about the insurance management systems and how-abouts of their working scenarios. These are some issues as listed by them:

- To the customers, who actually are alien to the back-end of the system, and only interact with the front-end, do not find it an easy-to-use site, i.e. it is difficult for them to keep track of their own policies and profile, their payment history and dues.
- To the administrators and staff, who have to operate and manage the system, it seems difficult to handle such a huge database and it becomes a tedious everyday job for them.
- It is difficult to track the history of a user and his/her assets and acquaintances.
- Each time a separate staff is assigned to a customer, which disrupts the continuity of conversation and flow of process.

Objective of this project is to simplify the insurance-related procedures for the end-users, administrators, staff as well as other stakeholders and resolve the various issues that they encounter in their day-to-day processes.

IDENTIFYING THE STAKEHOLDERS

The insurance management system has its connection with various people at different levels, some who are authorized to modify or alter the database, others who can view a selected part of the database, and remaining who have access to an even smaller segment of the database. We identified the following two levels to categorize the various stakeholders -

- FULL CONTROL (Full viewing as well as editing rights)
- PARTIAL CONTROL (Visibility and Access to a segment of database)

LEVEL 1 STAKEHOLDERS : FULL CONTROL

ADMINISTRATORS

These are the people, who have the entire rights over the database, for example, the manager of each branch, etc. They can add or remove employees and agents, change profiles, view benefits, premiums, change or alter policy details etc.

SHAREHOLDERS

These are the people who have a share in this particular insurance firm. In this particular case, lets assume, that the five members of our team hold some equity over this firm, and hence have full control of the entire database and activities of the firm.

LEVEL 2 STAKEHOLDERS : PARTIAL CONTROL

AGENTS

These are the people through which customers get to the insurance firm, i.e. the agents bring the customers, and in-turn get monetary commission. They have rights to add or remove their customers or alter some segments of data for their customers.

STAFF

These are the employees of the firm, i.e. their day to day activities include keeping track of the payments made by users, update user accounts, provide customer care services and assist the customers on-site with other essential services.

INDIVIDUAL USERS

These are the customers, who have enrolled in some policies offered by the firm. They can login and view their profile, edit it, avail services etc. These are the end-users of the application and do not have access to other segments of the database, apart from their own profile.

ORGANISATIONS

These are companies or institutions like Banks, Travel agencies or Airlines, Hospitals and Property Dealers, with whom the firm has collaborated for a particular duration. These organisations "promote" or "sell" the policies of the firm to their customers.

WEEK 2

USE CASE SCENARIOS FOR STAKEHOLDERS

ADMINISTRATORS

- How many policies has the firm registered in 2019?
- How has the growth rate been for a branch from last year?
- Which collaboration has resulted in maximum customers enrolled under their name?
- Which agent has bought the maximum customers?
- How many people work in a particular branch of the firm?

SHAREHOLDERS

- How many branches does the firm have?
- How much revenue is being used in the salaries of employees?
- How many medical insurances have been registered in 2019?
- How many people claimed their insurance amount last year?
- Should the policy duration be decreased to increase users?

AGENTS

- I wish to add a customer under my ID
- How many customers have I enrolled ?
- What is my commission factor?
- How much do I earn through this firm monthly?
- I wish to update my account details like phone number

STAFF

- Reviewing customer's history
- Add or remove a customer's database
- Update the coverage amount and other details for a customer
- Review agents account and manage their commissions
- Assist customers online

INDIVIDUAL USERS

- I want to pay my monthly premium for the policy
- I wish to update my account details like phone number
- How do I seek assistance for something online?
- I wish to withdraw my policy
- I wish to claim my policy amount

ORGANISATIONS

- How many customers has the firm got via my company?
- I wish to view and manage my account on the site
- I wish to add customers that come via the organisation
- I wish to extend collaboration with the firm
- How much revenue has my organisation made by this collaboration?

NOTE : THESE ARE THE TENTATIVE USE CASE SCENARIOS. THERE WILL BE MANY MORE USE CASES ADDED TO THESE, AND SOME MIGHT BE DELETED, WHILE WORKING THROUGH THE PROJECT .

BASIC ENTITIES AND RELATIONSHIPS

CLIENT TABLE

this database will contain entire details of the clients, like phone number, aadhar number etc. This is the complete list of database of clients, which includes individual customers as well as institutions who have enrolled for a policy at the firm. Hence this can be split innto two.

POLICY TABLE

This has a detailed database of all the policies offered by the firm, along with their policy ID, description, duration etc. This does not have anything to concern with the customers, this is just a list of offered policies.

AGENT TABLE

This database has the complete data about every agent that brings customers to the firm, and what is the commission factor, how many customers has an agent enrolled etc.

STAFF TABLE

This is the database of employees of the firm and includes details like staff ID, their branch ID etc. This list can be split into various other tables on the basis of position - admin, service staff etc.

BRANCH TABLE

This contains the details of the branch of the firm like number of employees, registered policies, branch ID etc.

LOGIN TABLE

This contains the login details for every stakeholder, username, password, phone number and email ID.

INSURANCE TABLE

This is the master table for all insurance policies enrolled by clients and has details like duration, coverage amount, agent ID, staff ID, unique insurance ID etc. This is further split into 5 tables, one for each kind of policy.

COMPANY TABLE

This contains a list of all companies/organisations that have a collaboration with the firm, and includes details like registration number of company, number of enrolled customers, duration of collab etc.

NOTE : THESE ARE THE TENTATIVE ENTITIES IN OUR SYSTEM. THERE CAN BE ADDITIONS AND SUBTRACTIONS FROM THE LIST . CURRENTLY, THERE ARE APPROX. 14 TABLES INVOLVED IN THE SYSTEM, AS STATED ABOVE.

WEEK 3

DEVELOPING THE SCHEMA

CLIENTS		
ATTRIBUTES	**TYPE**	**CONSTRAINTS**
client_name	varchar(30)	NOT NULL
client_ph	char(10)	NOT NULL UNIQUE
client_email	varchar(20)	NOT NULL UNIQUE
client_aadhar	char(12)	NOT NULL UNIQUE
client_PAN	varchar(10)	NOT NULL UNIQUE
client_ID	char(12)	NOT NULL UNIQUE
agent_ID	char(12)	
client_DOB	DATE	
client_sex	char(1)	
company_reg_no	varchar(20)	
branch_ID	char(10)	

Primary Key : client_ID
Foreign Keys : agent_ID references AGENTS
branch_ID references BRANCH
company_reg_no references COMPANIES

POLICIES		
ATTRIBUTES	**TYPE**	**CONSTRAINTS**
policy_key	char(6)	NOT NULL UNIQUE
ins_type	varchar(7)	NOT NULL CHECK(ins_type IN('Life', 'Home', 'Travel', 'Vehicle', 'Medical'))
premium	MEDIUMINT	
eligibility_cond	varchar(100)	NOT NULL
ppm	FLOAT(15,4)	NOT NULL
coverage_amt	FLOAT(15,4)	NOT NULL
duration	TINYINT	NOT NULL
terms_conditions	varchar(100)	NOT NULL
policy_name	varchar(30)	NOT NULL

Primary Key : policy_key

DEVELOPING THE SCHEMA

AGENTS

ATTRIBUTES	TYPE	CONSTRAINTS
agent_name	varchar(30)	NOT NULL
agent_ph	char(10)	NOT NULL UNIQUE
agent_email	varchar(20)	NOT NULL UNIQUE
agent_aadhar	char(12)	NOT NULL UNIQUE
agent_ID	char(12)	NOT NULL UNIQUE
commission_factor	FLOAT(4,2)	NOT NULL

Primary Key : agent_id

STAFF

ATTRIBUTES	TYPE	CONSTRAINTS
employee_name	varchar(30)	NOT NULL
employee_ph	char(10)	NOT NULL UNIQUE
employee_email	varchar(20)	NOT NULL UNIQUE
employee_aadhar	char(12)	NOT NULL UNIQUE
employee_PAN	char(10)	NOT NULL UNIQUE
employee_ID	char(12)	NOT NULL UNIQUE
branch_ID	char(10)	NOT NULL
department	varchar(15)	
position	varchar(15)	
salary	FLOAT(15, 4)	NOT NULL

Primary Key : employee_ID
Foreign Key: branch_ID references BRANCH

LOGIN

ATTRIBUTES	TYPE	CONSTRAINTS
email	varchar(20)	NOT NULL UNIQUE
username	varchar(12)	NOT NULL UNIQUE
password	varchar(16)	NOT NULL UNIQUE
user_type	varchar(15)	NOT NULL

Primary Key : username

DEVELOPING THE SCHEMA

BRANCH

ATTRIBUTES	TYPE	CONSTRAINTS
branch_name	varchar(30)	NOT NULL
branch_ph	char(10)	NOT NULL UNIQUE
branch_email	varchar(20)	NOT NULL UNIQUE
branch_ID	char(10)	NOT NULL UNIQUE

Primary Key : branch_ID

TRANSACTIONS

ATTRIBUTES	TYPE	CONSTRAINTS
transaction_ID	char(16)	NOT NULL UNIQUE
Unique_ins_ID	char(12)	NOT NULL
amount	FLOAT(13,5)	NOT NULL
payment_datetime	DATETIME	NOT NULL

Primary Key : transaction_ID
Foreign Key : Unique_Ins_ID references INSURANCES

INSURANCES

ATTRIBUTES	TYPE	CONSTRAINTS
policy_key	char(6)	NOT NULL
client_ID	char(12)	NOT NULL
start_date	DATE	NOT NULL
branch_ID	char(10)	NOT NULL
Unique_Ins_ID	char(12)	NOT NULL UNIQUE
dues	FLOAT(15,4)	

Primary Key : Unique_Ins_ID
Foreign Key : policy_key references POLICIES
client_ID references CLIENTS
branch_ID references BRANCH

DEVELOPING THE SCHEMA

	HOME_INSURANCE		
	ATTRIBUTES	**TYPE**	**CONSTRAINTS**
	Unique_Ins_ID	char(12)	NOT NULL UNIQUE
	prop_location	varchar(40)	NOT NULL UNIQUE
	path_to_prop_papers	varchar(60)	NOT NULL UNIQUE
	prop_area	MEDIUMINT	NOT NULL
	owners_names	varchar(40)	NOT NULL

Primary Key : Unique_Ins_ID
Foreign Key : Unique_Ins_ID references INSURANCES

	LIFE_INSURANCE		
	ATTRIBUTES	**TYPE**	**CONSTRAINTS**
	Unique_Ins_ID	char(12)	NOT NULL UNIQUE
	nominee_name1	varchar(20)	NOT NULL
	nominee_name2	varchar(20)	NOT NULL
	path_to_birth_certificate	varchar(60)	NOT NULL

Primary Key : Unique_Ins_ID
Foreign Key : Unique_Ins_ID references INSURANCES

DEVELOPING THE SCHEMA

TRAVEL_INSURANCE		
ATTRIBUTES	**TYPE**	**MEDICAL**
Unique_Ins_ID	char(12)	NOT NULL UNIQUE
travel_date	DATE	NOT NULL
travel_type	varchar(4)	NOT NULL CHECK(travel_type in ('Air', 'Rail', 'Road'))
travel_details	varchar(100)	NOT NULL

Primary Key : Unique_Ins_ID
Foreign Key : Unique_Ins_ID references INSURANCES

MEDICAL_INSURANCE		
ATTRIBUTES	**TYPE**	**CONSTRAINTS**
Unique_Ins_ID	char(12)	NOT NULL UNIQUE
medical_history	varchar(100)	NOT NULL
path_to_medical_bills	varchar(60)	NOT NULL UNIQUE

Primary Key : Unique_Ins_ID
Foreign Key : Unique_Ins_ID references INSURANCES

VEHICLE_INSURANCE		
ATTRIBUTES	**TYPE**	**CONSTRAINTS**
vehicle_ID	varchar(10)	NOT NULL
vehicle_type	varchar(10)	NOT NULL
Unique_Ins_ID	char(12)	NOT NULL UNIQUE
RC_num	varchar(20)	NOT NULL UNIQUE
Path_to_RC	varchar(60)	NOT NULL UNIQUE

Primary Key : Unique_Ins_ID
Foreign Key : Unique_Ins_ID references INSURANCES

DEVELOPING THE SCHEMA

(PART OF BONUS IMPLEMENTATION)

SHAREHOLDERS		
ATTRIBUTES	**TYPE**	**CONSTRAINTS**
share_name	varchar(20)	NOT NULL
equity_percentage	float(4,2)	NOT NULL
share_ID	char(6)	NOT NULL UNIQUE
share_email	varchar(20)	NOT NULL UNIQUE

Primary Key : share_ID

COMPANIES		
ATTRIBUTES	**TYPE**	**CONSTRAINTS**
company_name	varchar(30)	NOT NULL
company_reg_no	varchar(20)	NOT NULL UNIQUE
company_email	varchar(20)	NOT NULL UNIQUE
collab_duration	TINYINT	NOT NULL
company_ph	char(10)	NOT NULL UNIQUE
collab_start_date	DATE	NOT NULL
company_ID	char(12)	NOT NULL UNIQUE

Primary Key : company_ID

OFFERS		
ATTRIBUTES	**TYPE**	**CONSTRAINTS**
offer_ID	varchar(6)	NOT NULL UNIQUE
offer_desc	varchar(100)	NOT NULL
discount_factor	SMALLINT	NOT NULL
active	bool	NOT NULL
company_id	char(12)	NOT NULL

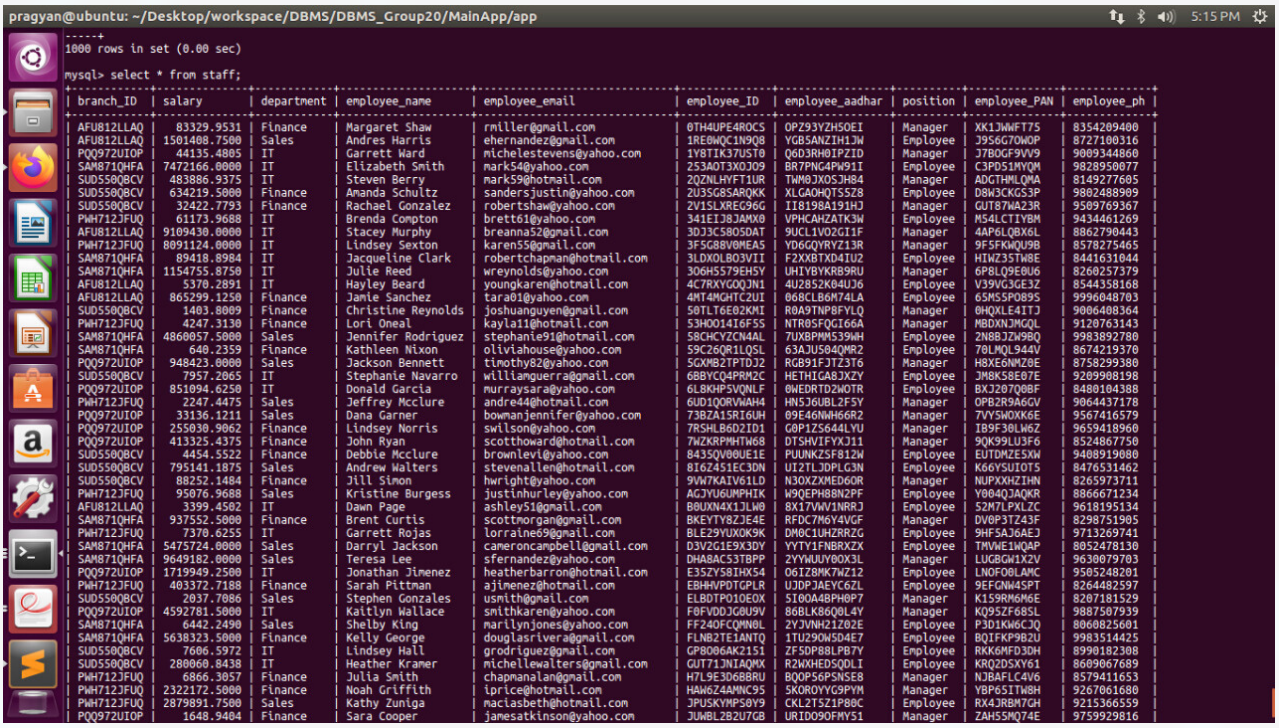
Primary Key : offer_ID
Foreign Key : company_ID references COMPANIES

WEEK 4

POPULATING THE TABLES

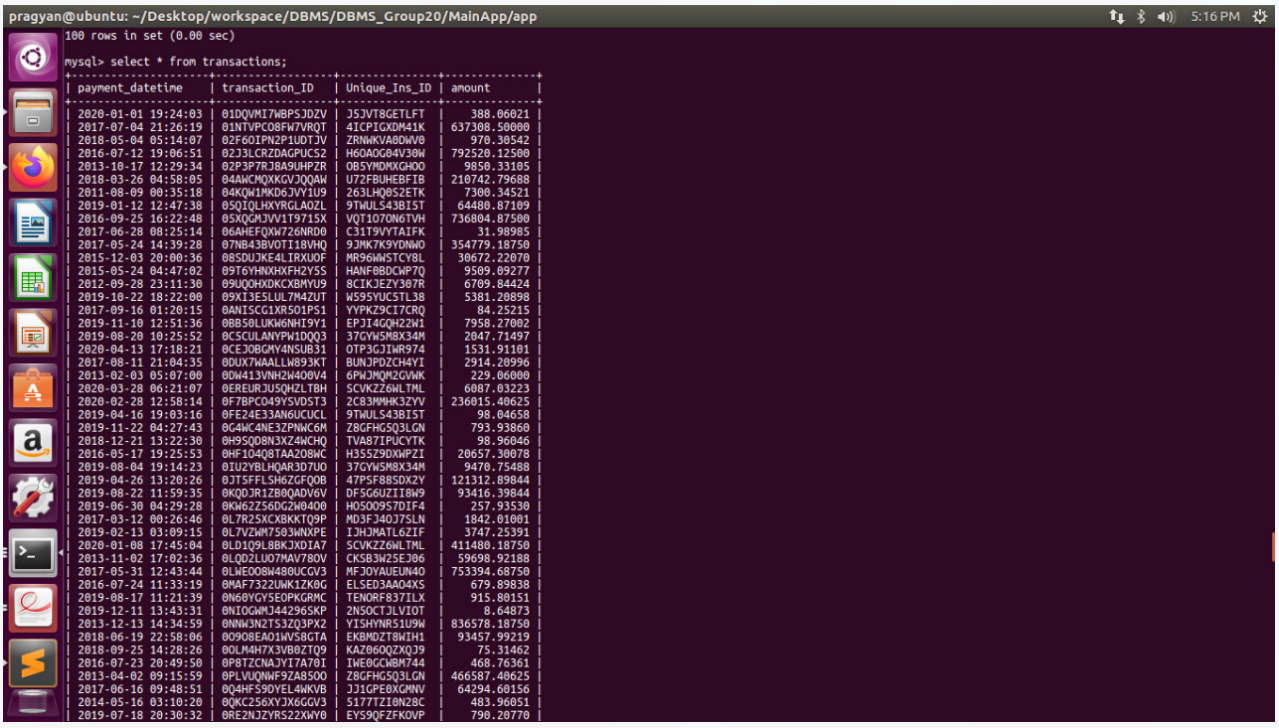
Here are some screenshots, that display the tables individually.

staff table



branch_ID	salary	department	employee_name	employee_email	employee_ID	employee_aadhar	position	employee_PAN	employee_ph
AFU812LLAQ	83329.9531	Finance	Margaret Shaw	rmiller@gmail.com	0TH4UPE4ROCS	OPZ93YZH5OEI	Manager	XK1JWWFT7S	8354209400
AFU812LLAQ	1501408.7500	Sales	Andres Harris	ehernandez@gmail.com	1RE0WQC1N9Q8	YGB5ANZIH1JW	Employee	J9S6G7OWOP	8727100316
PQQ972UIOP	44135.4805	IT	Garrett Ward	michelestevens@yahoo.com	1V8TIK37UST0	Q6D3RH0IPZID	Manager	J7BOGF9VV9	9009344860
SAM871QHFA	7472166.0000	IT	Elizabeth Smith	mark54@yahoo.com	253AOT3XOJO9	BR7PNG4PW9II	Employee	C3PD51MYQM	9828950877
SUD550QBCV	483886.9375	IT	Steven Berry	mark59@hotmail.com	2QZNLHYFT1UR	TWM0JXOSJHB4	Manager	ADGTHMLQWA	8149277605
SUD550QBCV	634219.5000	Finance	Amanda Schultz	sandersjustin@yahoo.com	2U3SG8SARQKK	XLGAOHQTSSZ8	Employee	D8K3CKGS3P	9802488909
SUD550QBCV	32422.7793	Finance	Rachael Gonzalez	robertshaw@yahoo.com	2V1SLXREG96G	II8198A191HJ	Manager	GUT87WA23R	9509769367
PWH712JFUQ	61173.9688	IT	Brenda Compton	brett61@yahoo.com	341EIJ8JAMX0	VPHCAHZATK3W	Employee	M54LCTIYBM	9434461269
AFU812LLAQ	9109430.0000	IT	Stacey Murphy	breanna52@gmail.com	3DJ3CS8OSDAT	9UCL1VO2GI1F	Manager	4AP6LQBX6L	8862790443
PWH712JFUQ	8091124.0000	IT	Lindsey Sexton	karen55@gmail.com	3F5G88V0MEAS	YD6GQYRYZ13R	Manager	9F5FKWQU9B	8578275465
SAM871QHFA	89418.8984	IT	Jacqueline Clark	robertchapman@hotmail.com	3LDXOLBO3VII	F2XXBTXD4IU2	Employee	HIWZ3STW8E	8441631044
SAM871QHFA	1154755.8750	IT	Julie Reed	wreynolds@yahoo.com	3O6H5S79EH5Y	UHIYBYKRB9RU	Manager	6P8LQ9E0U6	8260257379
AFU812LLAQ	5370.2891	IT	Hayley Beard	youngkaren@hotmail.com	4C7RXYGOQJN1	4U2852K04UJ6	Employee	V39VG3GE3Z	8544358168
AFU812LLAQ	865299.1250	Finance	Jamie Sanchez	tara01@yahoo.com	4MT4MGHTC2UI	068CLB6M74LA	Employee	65MSSPO89S	9996048703
SUD550QBCV	1403.8009	Finance	Christine Reynolds	joshuanguyen@gmail.com	50TLT6E02KMI	R0A9TNP8FYLQ	Manager	0HQXLE4ITJ	9006408364
PWH712JFUQ	4247.3130	Finance	Lori Oneal	kayla11@hotmail.com	53HOO4II6F5S	NTR0SFQGI66A	Manager	MBDXNJMGQL	9120763143
SAM871QHFA	4860057.5000	Sales	Jennifer Rodriguez	stephanie91@hotmail.com	58CHCYZCN4AL	7UXBPMMS39WH	Employee	2N8BJZW9BQ	9983892780
SAM871QHFA	640.2359	Finance	Kathleen Nixon	oliviahouse@yahoo.com	59C26QR1LQSL	63AJU504QMR2	Employee	70LMQL944V	8674219370
PQQ972UIOP	948423.0000	Sales	Jackson Bennett	timothy82@yahoo.com	5GXMB2TPTDJ2	RGB91FJT23T6	Manager	H8XE6NMZ0E	8758299380
SUD550QBCV	7957.2065	IT	Stephanie Navarro	williamguerra@gmail.com	6BBYCQ4PRM2C	HETHIGA8JXZY	Employee	JM8K58E0TE	9209908198
PQQ972UIOP	851094.6250	IT	Donald Garcia	murraysara@yahoo.com	6L8KHP5VQNLF	0WEDRTD2WOTR	Employee	BXJ207Q0BF	8480104388
PWH712JFUQ	2247.4475	Sales	Jeffrey Mcclure	andre44@hotmail.com	6UD1QORVWAH4	HN5J6UBL2F5Y	Manager	OPB2R9A6CV	9064437178
PQQ972UIOP	33136.1211	Sales	Dana Garner	bowmanjennifer@yahoo.com	73BZA15RI6UH	09E46NWH6GR2	Manager	7VY5WOXK6E	9567416579
PQQ972UIOP	255030.9062	Finance	Lindsey Norris	swilson@yahoo.com	7RSHLB6D2ID1	G0P1ZS644LYU	Manager	IB9F30LW6Z	9659418960
PQQ972UIOP	413325.4375	Finance	John Ryan	scotthoward@hotmail.com	7WZKRPMHTW68	DTSHVIFYXJ11	Manager	9QK99LU3F6	8524867750
SUD550QBCV	4454.5522	Finance	Debbie Mcclure	brownlevi@yahoo.com	843SQV00UE1E	PUUNKZSF812W	Employee	EUTDMZE5XW	9408919080
SUD550QBCV	795141.1875	Sales	Andrew Walters	stevenallen@hotmail.com	8I6Z4S1EC3DN	UI2TLJDPLG3N	Employee	K66YSUIOTS	8476531462
SUD550QBCV	88252.1484	Finance	Jill Simon	hwright@yahoo.com	9VW7KAIV61LD	N3OXZXMED6OR	Manager	NUPXXHZIHN	8265973711
PWH712JFUQ	95076.9688	Sales	Kristine Burgess	justinhurley@yahoo.com	AGJVU6UMPHIK	W9QEPH88N2PF	Employee	Y004QJAQKR	8866671234
AFU812LLAQ	3399.4502	IT	Dawn Page	ashley51@gmail.com	B0UXN4X1JLW0	8X17VWV1NRRJ	Employee	52M7LPXLZC	9618195134
SAM871QHFA	937552.5000	Finance	Brent Curtis	scottmorgan@gmail.com	BKEYTY8ZJE4E	RFDC7M6Y4VGF	Manager	DV0P3TZ43F	8298751905
PWH712JFUQ	7370.6255	IT	Garrett Rojas	lorraine69@gmail.com	BLE29YUXOK9K	DM0C1UHZRRZG	Employee	9HF5AJ6AEJ	9713269741
SAM871QHFA	5475724.0000	Sales	Darryl Jackson	cameroncampbell@gmail.com	D3V2G1E9X3DY	YYTY1FNBRXZX	Employee	TMVWE1WQAP	8052478130
SAM871QHFA	9649182.0000	Sales	Teresa Lee	sfernandez@yahoo.com	DHA8AC53TBPP	2YYWUUY0OX3L	Manager	LUGBGW1X2V	9630079703
PQQ972UIOP	1719949.2500	IT	Jonathan Jimenez	heatherbarron@hotmail.com	E35ZYS8IHX54	O6IZ8MK7WZ12	Employee	LNQFO0LAMC	9505248201
PWH712JFUQ	403372.7188	Finance	Sarah Pittman	ajimenez@hotmail.com	EBHVPDTGPLR	UJDPJAEYC6ZL	Employee	9EFGNW4SPT	8264482597
SUD550QBCV	2037.7086	Sales	Stephen Gonzales	usmith@gmail.com	ELBDTPO1OEOX	5I0OA4BPH0P7	Manager	K159RM6M6E	8207181529
PQQ972UIOP	4592781.5000	IT	Kaitlyn Wallace	smithkaren@yahoo.com	F0FVDDJG0U9V	86BLK86Q0L4Y	Manager	KQ95ZF68SL	9887507939
SAM871QHFA	6442.2490	Sales	Shelby King	marilynjones@yahoo.com	FF24OFCQMN0L	2YJVNH21Z02E	Employee	P3D1KW6CJQ	8060825601
SAM871QHFA	5638323.5000	Finance	Kelly George	douglasrivera@gmail.com	FLNB2TE1ANTQ	1TU29OWSD4E7	Employee	BQIFKP9B2U	9983514425
SUD550QBCV	7606.5972	IT	Lindsey Hall	grodriguez@gmail.com	GP8O06AK2151	ZF5DP88LPB7Y	Employee	RKK6MFD3DH	8990182308
SUD550QBCV	280060.8438	IT	Heather Kramer	michellewalters@gmail.com	GUT71JNIAQMX	R2WXHEDSQDLI	Employee	KRQ2DSXY61	8609067689
PWH712JFUQ	6866.3057	Finance	Julia Smith	chapmanalan@gmail.com	H7L9E3D6BBRU	BQOPS6PSNSE8	Manager	NJBAFLC4V6	8579411653
PWH712JFUQ	2322172.5000	Finance	Noah Griffith	iprice@hotmail.com	HAW6Z4AMNC95	5KOROYYG9PYM	Manager	YBP6SITW8H	9267061680
PWH712JFUQ	2879891.7500	Sales	Kathy Zuniga	maciasbeth@hotmail.com	JPUSKYMPS0Y9	CKL2TSZ1P80C	Employee	RX4JRBM7GH	9215366559
PQQ972UIOP	1648.9404	Finance	Sara Cooper	jamesatkinson@yahoo.com	JUWBL2B2U7GB	URIDO9OFMY51	Manager	ZAH55MQ74E	9759929816

transactions table



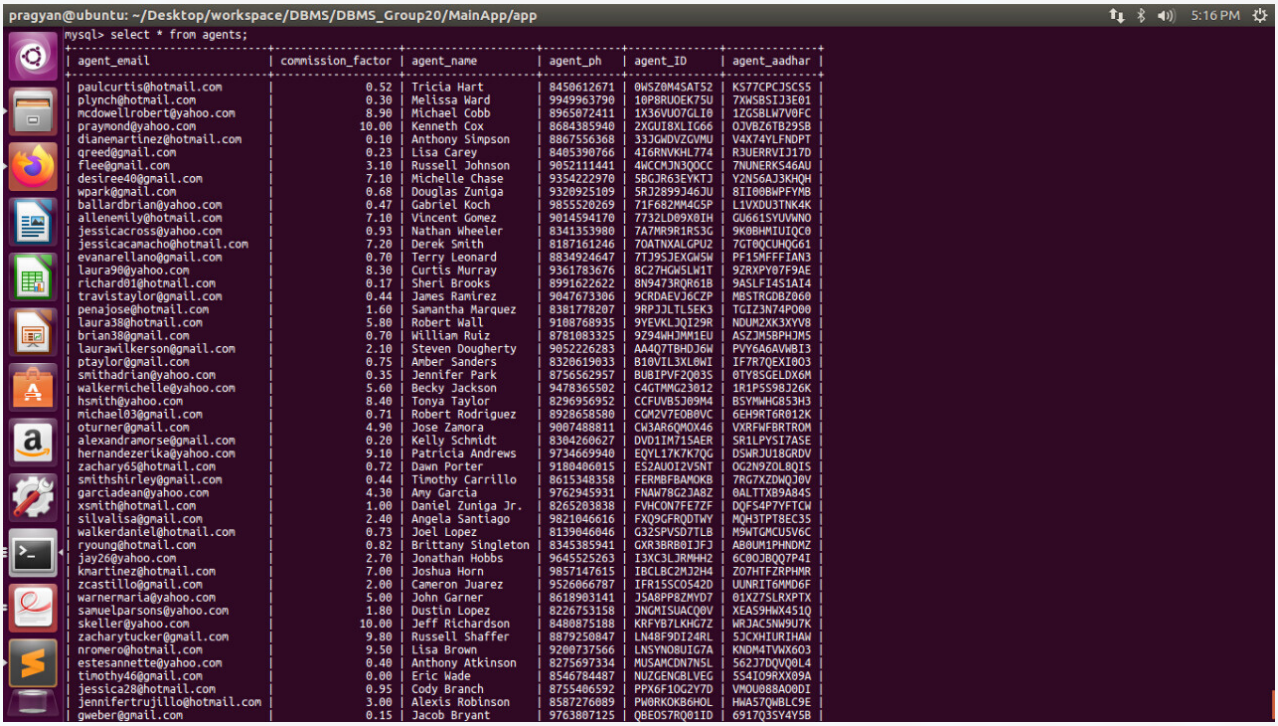
payment_datetime	transaction_ID	Unique_Ins_ID	amount
2020-01-01 19:24:03	01DQVMI7WBPSJDZV	J5JVT8GETLFT	388.06021
2017-07-04 21:26:19	01NTVPC08FW7VRQT	4ICPIGXDM41K	637308.50000
2018-05-04 05:14:07	02F6OIPN2P1UDTJV	ZRNWKVA8OWV0	970.30542
2016-07-12 19:06:51	02J3LCRZDAGPUCS2	H6OAOG04V30W	792520.12500
2013-10-17 12:29:34	02P3P7RJ8A9UAPZR	OBSYMDMXGHOO	9850.33105
2018-03-26 04:58:05	04AWCMQXKGVJQQAW	U72FBUHEBFIB	210742.79688
2011-08-09 00:35:18	04KQW1MKD6JVY1U9	263LHQ0S2ETK	7300.34521
2019-01-12 12:47:38	05QIQLHXYRGLAOZL	9TWULS43BI5T	64480.87109
2016-09-25 16:22:48	05XQGMJVV1T9715X	VQT107ON6TVH	736804.87500
2017-06-28 08:25:14	06AHEFQXW726NRD0	C31T9VYTAIFK	31.98985
2017-05-24 14:39:28	07NB43BVOTI18VHQ	9JMK7K9YDNWO	354779.18750
2015-12-03 20:00:36	08SDUJKE4LIRXUOF	MR96WWSTCY8L	30672.22070
2015-05-24 04:47:02	09T6YHNXHXFH2Y5S	HANF0BDCWP7Q	9509.09277
2012-09-28 23:11:30	09UQOHXDKCXBMYU9	8CIKJEZY307R	6709.84424
2019-10-22 18:22:00	09XI3E5LUL7M4ZUT	W595YUC5TL38	5381.20898
2017-09-16 01:20:15	0ANISCG1XR5O1PS1	YYPKZ9CI7CRQ	84.25215
2019-11-10 12:51:36	0BB50LUKW6NHI9Y1	EPJI4GQH22W1	7958.27002
2019-08-20 10:25:52	0CSCULANYPW1DQQ3	37GYW5M8X34M	2047.71497
2020-04-13 17:18:21	0CEJOBGMY4NSUB31	OTP3GJIWR974	1531.91101
2017-08-11 21:04:35	0DUX7WALLK893KT	BUNJPDZCH4YI	2914.20996
2013-02-03 05:07:00	0DW413VNHZW4O0V4	6PWJMQM2GVWK	229.06000
2020-03-28 06:21:07	0EREURJUSQHZLTBH	SCVKZZ6WLTML	6087.03223
2020-02-28 12:58:14	0F7BPCO49YSVDST3	2C83MMHK3ZYV	236015.40625
2019-04-16 19:03:16	0FE24E33AN6UCUCL	9TWULS43BI5T	98.04658
2019-11-22 04:27:43	0G4WC4NE3ZPNWC6M	Z8GFHGSQ3LGN	793.93860
2018-12-21 13:22:30	0H9SQD8N3XZ4WCHQ	TVA8TIPUCYTK	98.96046
2016-05-17 19:25:53	0HF1O4Q8TAA2O8WC	H355Z9DXWPZI	20657.30078
2019-08-04 19:14:23	0IU2YBLHQAR3D7UO	37GYW5M8X34M	9470.75488
2019-04-26 13:20:26	0JT5FFLSH6ZGFQOB	47PSF88SDX2Y	121312.89844
2019-08-22 11:59:35	0KQDJR1ZB0QADV6V	DF5G6UZII8W9	93416.39844
2019-06-30 04:29:28	0KW62Z56DG2W04O0	HO5OO9S7DIF4	257.93530
2017-03-12 00:26:46	0L7R25XCXBKKTQ9P	MD3FJ4OJ75LN	1842.01001
2019-02-13 03:09:15	0L7VZWM750JWNXPE	IJHJMATL6ZIF	3747.25391
2020-01-08 17:45:04	0LD1Q9L8BKJXDIA7	SCVKZZ6WLTML	411480.18750
2013-11-02 17:02:36	0LQD2LUO7MAV78OV	CKSB3W25EJ06	59698.92188
2017-05-31 12:43:44	0LWEOO8W480UCGV3	MFJOYAUEUN4O	753394.68750
2016-07-24 11:33:19	0MAF7322UWK1ZK0G	ELSED3AAO4XS	679.89838
2019-08-17 11:21:39	0N60YGYSEOPKGRMC	TENORF837ILX	915.80151
2019-12-11 13:43:31	0NIOGWMJ44296SKP	2N5OCTJLVIOT	8.64873
2013-12-13 14:34:59	0NNW3NZTS3ZQ3PX2	YI5HYNR51U9W	836578.18750
2018-06-19 22:58:06	0O9O8EAO1WVS8GTA	EKBMDZT8WIH1	93457.99219
2018-09-25 14:28:26	0OLM4H7X3VB0ZTQ9	KAZ06OQZXQJ9	75.31462
2016-07-23 20:49:50	0P8TZCNAJYI7A70I	IWE0GCWBM744	468.76361
2013-04-02 09:15:59	0PLVUQNWF9ZA85OO	Z8GFHGSQ3LGN	466587.40625
2017-06-16 09:48:51	0Q4HFS9DYEL4WKVB	JJ1GPE0XGMNV	64294.60156
2014-05-16 03:10:20	0QKC256XYJX6GGV3	5177TZI0N28C	483.96051
2019-07-18 20:30:32	0RE2NJZYRS22XWY0	EYS9QFZFKOVP	790.20770

NOTE: Screenshots for other tables haven't been attached here.

POPULATING THE TABLES

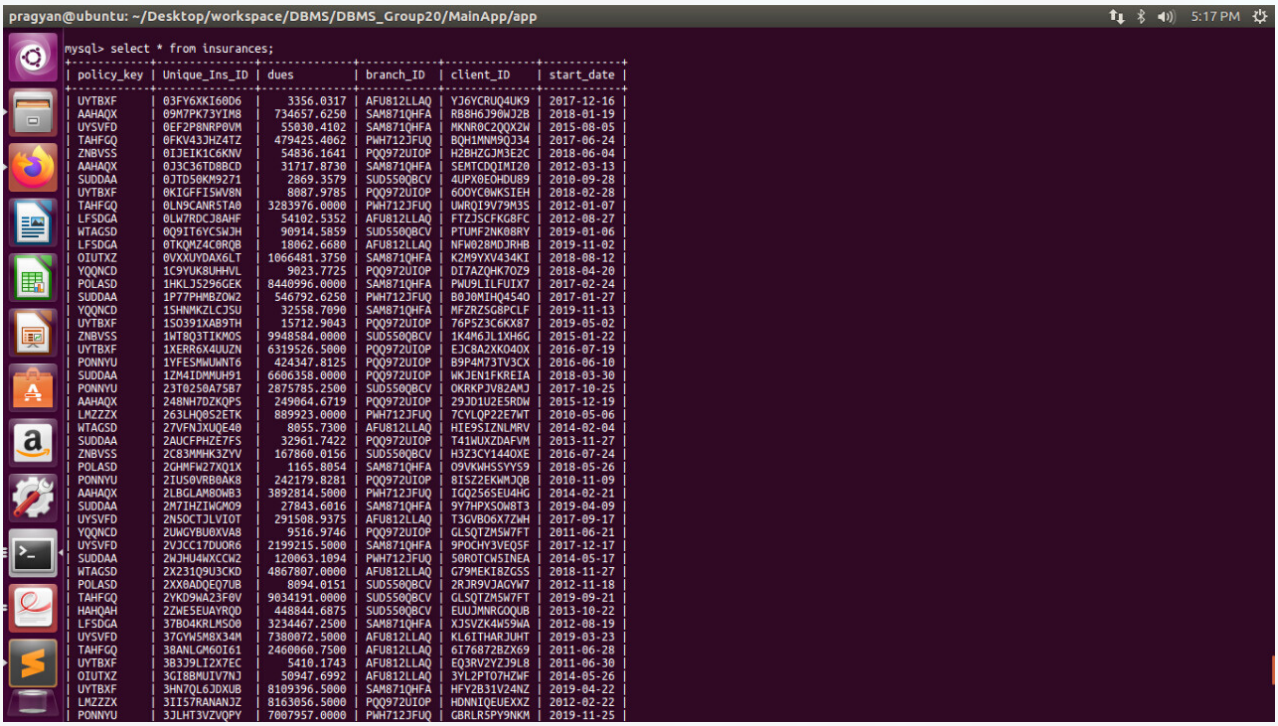
Here are some screenshots, that display the tables individually.

agents table



agent_email	commission_factor	agent_name	agent_ph	agent_ID	agent_aadhar
paulcurtis@hotmail.com	0.52	Tricia Hart	8450612671	0WSZ0M4SAT52	KS77CPCJSCS5
plynch@hotmail.com	0.30	Melissa Ward	9949963790	10P8RUOEK75U	7XWSBSIJ3E01
mcdowellrobert@yahoo.com	8.90	Michael Cobb	8965072411	1X36VUO7GLI0	1ZGSBLW7V0FC
praymond@yahoo.com	10.00	Kenneth Cox	8684385940	2XGUI8XLIG66	OJVBZ6TB29SB
dianemartinez@hotmail.com	0.10	Anthony Simpson	8867556368	33JGWDVZGVMU	V4X74YLFNDPT
qreed@gmail.com	0.23	Lisa Carey	8405390766	4I6RNVKHL774	R3UERRVIJ17D
flee@gmail.com	3.10	Russell Johnson	9052111441	4WCCMJN3QOCC	7NUNERKS46AU
desiree40@gmail.com	7.10	Michelle Chase	9354222970	5BGJR63EYKTJ	Y2N56AJ3KHQH
wpark@gmail.com	0.68	Douglas Zuniga	9328925109	5RJ2899J46JU	8II00BWFFVMB
ballardbrian@yahoo.com	0.47	Gabriel Koch	9855520269	71F682MM4GSP	L1VXDU3TNK4K
allenemily@hotmail.com	7.10	Vincent Gomez	9014594170	7732LD09X0IH	GU661SYUVWNO
jessicacross@yahoo.com	0.93	Nathan Wheeler	8341353980	7A7MR9R1RS3G	9K0BHMIUIQC0
jessicacamacho@hotmail.com	7.20	Derek Smith	8187161246	7OATNXALGPU2	7GT0QCUHQG61
evanarellano@gmail.com	0.70	Terry Leonard	8834924647	7TJ9SJEXGW5W	PF15MFFFIAN3
laura90@yahoo.com	8.30	Curtis Murray	9361783676	8C27HGW5LW1T	9ZRXPY07F9AE
richard01@hotmail.com	0.17	Sheri Brooks	8991622622	8N9473RQR61B	9ASLFI4S1AI4
travistaylor@gmail.com	0.44	James Ramirez	9047673306	9CRDAEVJ6CZP	MBSTRGDBZ060
penajose@hotmail.com	1.60	Samantha Marquez	8381778207	9RPJJLTL5EK3	TGIZ3N74PO00
laura38@hotmail.com	5.80	Robert Wall	9108768935	9YEVKLJQI29R	NDUM2XK3XYV8
brian38@gmail.com	0.70	William Ruiz	8781083325	9Z94WHJMM1EU	ASZJMSBPHJM5
laurawilkerson@gmail.com	2.10	Steven Dougherty	9052226283	AA4Q7TBHDJ6W	PVY6A6AVWBI3
ptaylor@gmail.com	0.75	Amber Sanders	8328619033	B10VIL3XL8WI	IF7R7QEXI8O3
smithadrian@yahoo.com	0.35	Jennifer Park	8756562957	BUBIPVF2Q83S	0TY8SGELDX6M
walkermichelle@yahoo.com	5.60	Becky Jackson	9478365502	C4GTMMG23812	1R1P5598J26K
hsmith@yahoo.com	8.40	Tonya Taylor	8296956952	CCFUVBSJ09M4	B5YMWKG853H3
michael03@gmail.com	0.71	Robert Rodriguez	8928658580	CGM2V7EOB0VC	6EH9RT6R012K
oturner@gmail.com	4.90	Jose Zamora	9807488811	CW3AR6QMOX46	VXRFWFBRTROM
alexandramorse@gmail.com	0.20	Kelly Schmidt	8304260627	DVD1IM71SAER	SR1LPYSI7ASE
hernandezerika@yahoo.com	9.10	Patricia Andrews	9734669940	EQYL17K7K7QG	DSWRJU18GRDV
zachary65@hotmail.com	0.72	Dawn Porter	9180406015	ES2AUOI2V5NT	OG2N9ZOL8QIS
smithshirley@gmail.com	0.44	Timothy Carrillo	8615348358	FERMBFBAMOKB	7RG7XZDWQJ0V
garciadean@yahoo.com	4.30	Amy Garcia	9762945931	FNAW78G2JA8Z	0ALTTXB9A84S
xsmith@hotmail.com	1.00	Daniel Zuniga Jr.	8265203838	FVHCON7FE7ZF	DQFS4P7YFTCW
silvalisa@gmail.com	2.40	Angela Santiago	9821046616	FXQ9GFRQDTWY	MQH3TPT8EC35
walkerdaniel@hotmail.com	0.73	Joel Lopez	8139046046	G32SPVSD7TLB	M9WTGMCU5V6C
ryoung@hotmail.com	0.82	Brittany Singleton	8345385941	GXR3BRB0IJFJ	AB0UM1PHNDMZ
jay26@yahoo.com	2.70	Jonathan Hobbs	9645525263	I3XC3LJRMHH2	6C0OJBQQ7P4I
kmartinez@hotmail.com	7.00	Joshua Horn	9857147615	IBCLBC2MJ2H4	ZO7HTFZRPHMR
zcastillo@gmail.com	2.00	Cameron Juarez	9526066787	IFR15SCO542D	UUNRIT6MMD6F
warnernaria@yahoo.com	5.00	John Garner	8618903141	J5ABPP8ZMYD7	01XZ7SLRXPTX
samuelparsons@yahoo.com	1.80	Dustin Lopez	8226753158	JNGM1SUACQ0V	XEAS9HWX451Q
skeller@yahoo.com	10.00	Jeff Richardson	8480875188	KRFYB7LKHG7Z	WRJACSNW9U7K
zacharytucker@gmail.com	9.80	Russell Shaffer	8879250847	LN48F9DI24RL	5JCXHIURIHAW
nromero@hotmail.com	9.50	Lisa Brown	9200737566	LNSYNOBUIG7A	KNDM4TVWX6O3
estesannette@yahoo.com	0.40	Anthony Atkinson	8275697334	MUSAMCDN7N5L	562J7DQVQ0L4
timothy46@gmail.com	0.00	Eric Wade	8546784487	NUZGENGBLVEG	5S4IO9RXX09A
jessica28@hotmail.com	0.95	Cody Branch	8755406592	PPX6F1OG2Y7D	VMOU088AO0DI
jennifertrujillo@hotmail.com	3.00	Alexis Robinson	8587276089	PW0RKOKB6HOL	HWAS7QWBLC9E
gweber@gmail.com	0.15	Jacob Bryant	9763807125	QBEOS7RQ01ID	6917Q3SY4Y5B

insurances table



policy_key	Unique_Ins_ID	dues	branch_ID	client_ID	start_date
UYTBXF	03FY6XKI60D6	3356.0317	AFU812LLAQ	YJ6YCRUQ4UK9	2017-12-16
AAHAQX	09M7PK73YIMB	734657.6250	SAM871QHFA	RB8H6J90WJ2B	2018-01-19
UYSVFD	0EF2P8NRP0VM	55030.4102	SAM871QHFA	MKNR0C2QQX2W	2015-08-05
TAHFGQ	0FKV43JHZ4TZ	479425.4062	PWH712JFUQ	BQH1MNM9QJ34	2017-06-24
ZNBVSS	0IJEIK1C6KNV	54836.1641	PQQ972UIOP	H2BHZGJM3E2C	2018-06-04
AAHAQX	0J3C36TD8BCD	31717.8730	SAM871QHFA	SEMTCDQIMI20	2012-03-13
SUDDAA	0JTD50KM9271	2869.3579	SUDS50QBCV	4UPX0EOHDU89	2010-09-28
UYTBXF	0KIGFFISWV8N	8087.9785	PQQ972UIOP	6OOYC0WKSIEH	2018-02-28
TAHFGQ	0LN9CANR5TA0	3283976.0000	PWH712JFUQ	UWRQI9V79M3S	2012-01-07
LFSDGA	0LW7RDCJ8AHF	54102.5352	AFU812LLAQ	FTZJSCFKG8FC	2012-08-27
WTAGSD	0Q9IT6YCSWJH	90914.5859	SUDS50QBCV	PTUMF2NK0BRY	2019-01-06
LFSDGA	0TKQMZ4C0RQB	18062.6680	AFU812LLAQ	NFW028MDJRHB	2019-11-02
OIUTXZ	0VXXUYDAX6LT	1066481.3750	SAM871QHFA	K2M9YXV434KI	2018-08-12
YQQNCD	1C9YUK8UHHVL	9023.7725	PQQ972UIOP	DI7AZQHK7OZ9	2018-04-20
POLASD	1HKLJ5296GEK	8440996.0000	SAM871QHFA	PWU9LILFUIX7	2017-02-24
SUDDAA	1P77PHMBZOW2	546792.6250	PWH712JFUQ	B0J0MIHQ4S4O	2017-01-27
YQQNCD	1SHNMKZLCJSU	32558.7090	SAM871QHFA	MFZRZSG8PCLF	2019-11-13
UYTBXF	1SO391XAB9TH	15712.9043	PQQ972UIOP	76P5Z3C6KX87	2019-05-02
ZNBVSS	1WTRQ3TIKMOS	9940584.0000	SUDS50QBCV	1K4M6JL1XH6G	2015-01-22
UYTBXF	1XERR6X4UUZN	6319526.5000	PQQ972UIOP	EJC8A2XKO4OX	2016-07-19
PONNYU	1YFESMWUNNT6	424347.8125	PQQ972UIOP	B9P4M73TV3CX	2016-06-10
SUDDAA	1ZM4IDMMUH91	6606358.0000	PQQ972UIOP	WKJEN1FKREIA	2018-03-30
PONNYU	23T0250A75B7	2875785.2500	SUDS50QBCV	OKRKPJV82AMJ	2017-10-25
AAHAQX	248NH7DZKQPS	249064.6719	PQQ972UIOP	29JD1U2E5RDW	2015-12-19
LMZZZX	263LHQ0S2ETK	889923.0000	PWH712JFUQ	7CYLQP22E7WT	2010-05-06
WTAGSD	27VFNJXUQE40	8055.7300	AFU812LLAQ	HIE9SIZNLMRV	2014-02-04
SUDDAA	2AUCFPHZE7FS	32961.7422	PQQ972UIOP	T41WUXZDAFVM	2013-11-27
ZNBVSS	2C83MMHK3ZYV	167860.0156	SUDS50QBCV	H3Z3CY144OXE	2016-07-24
POLASD	2GHMFW27XQ1X	1165.8054	SAM871QHFA	O9VKWHSSYYS9	2018-05-26
PONNYU	2IUS0VRB0AK8	242179.8281	PQQ972UIOP	8ISZ2EKWMJQB	2010-11-09
AAHAQX	2LBGLAM8OWB3	3892814.5000	PWH712JFUQ	IGQ2S6SEU4HG	2014-02-21
SUDDAA	2M7IHZIWGMO9	27843.6016	SAM871QHFA	9Y7HPXSOW8T3	2019-04-09
UYSVFD	2NSOCTJLVIOT	291508.9375	AFU812LLAQ	T3GVBO6X7ZWH	2017-09-17
YQQNCD	2UWGYBU0XVA8	9516.9746	PQQ972UIOP	GLSQTZMSW7FT	2011-06-21
UYSVFD	2V2CC17DUOR6	2199215.5000	SAM871QHFA	9POCHY3VEQ5F	2017-12-17
SUDDAA	2WJHU4WXCCW2	120063.1094	PWH712JFUQ	50ROTCW5INEA	2014-05-17
WTAGSD	2X231Q9U3CKD	4867807.0000	AFU812LLAQ	G79MEKI8ZGSS	2018-11-27
POLASD	2XX0ADQEQ7UB	8094.0151	SUDS50QBCV	2RJR9VJAGYW7	2012-11-18
TAHFGQ	2YKD9WA23F0V	9034191.0000	SUDS50QBCV	GLSQTZMSW7FT	2019-09-21
HAHQAH	2ZWE5EUAYRQD	448844.6875	SUDS50QBCV	EUUJMNRGOQUB	2013-10-22
LFSDGA	37BO4KRLMSO0	3234467.2500	SAM871QHFA	XJSVZK4W59WA	2012-08-19
UYSVFD	37GYW5M8X34M	7380072.5000	AFU812LLAQ	KL6ITHARJUHT	2019-03-23
TAHFGQ	38ANLGM6OI61	2460060.7500	AFU812LLAQ	6I76872BZX69	2011-06-28
UYTBXF	3B3J9LI2X7EC	5410.1743	AFU812LLAQ	EQ3RV2YZJ9L8	2011-06-30
OIUTXZ	3GI8BMUIV7NJ	50947.6992	AFU812LLAQ	3YL2PTO7HZWF	2014-05-26
UYTBXF	3HN7QL6JDXUB	8109396.5000	SAM871QHFA	HFY2B31V24NZ	2019-04-22
LMZZZX	3II57RANANJZ	8163056.5000	PQQ972UIOP	HDNNIQEUEXXZ	2012-02-22
PONNYU	3JLHT3VZVQPY	7007957.0000	PWH712JFUQ	GBRLR5PY9NKM	2019-11-25

NOTE: Screenshots for other tables haven't been attached here.

WEEK 6

IDENTIFYING ATTRIBUTES FOR INDEXING

Indexing is used to speed up data retrieval from a table, by maintaining a pointer. Hence the attribute to be accessed the most often, which can be used to identify tuple, is chosen to be an idex for that table.

The identified indexes in each of the tables are as follows -

TABLE	INDEX
CLIENTS	client_ID
POLICIES	policy_key
AGENTS	agent_ID
STAFF	employee_ID
BRANCH	branch_ID
INSURANCE	Unique_Ins_ID, policy_key, client_ID
COMPANIES	company_ID, company_reg_no
OFFERS	offer_ID
TRANSACTIONS	transaction_ID, Unnique_ins_ID
LOGIN	username, email
SHAREHOLDER	share_ID
LIFE_INSURANCE	Unique_Ins_ID
HOME_INSURANCE	Unique_Ins_ID
VEHICLE_INSURANCE	Unique_Ins_ID
TRAVEL_INSURANCE	Unique_Ins_ID
MEDICAL_INSURANCE	Unique_Ins_ID

RELATIONAL QUERIES FOR BASIC FEATURES FOR EACH STAKEHOLDER

- VIEW ALL POLICIES

$$\Pi_{ins\_type,policy\_name,policy\_key,coverage\_amt,premium,duration,eligibility\_cond,terms\_conditions}(\sigma(policies))$$

- VIEW ALL POLICIES SOLD BY AN AGENT

$$\Pi_{B.client\_name,B.client\_ph,B.client\_email,C.ins\_type,A.start\_date,C.duration}(\sigma_{C.policy\_key=A.policy\_keyANDB.agent\_ID=xANDA.client\_ID=B.client\_ID}(\rho_A(\text{insurances}) \times\rho_B(\text{clients}) \times\rho_c(\text{policies})))$$

- VIEW ALL TRANSACTIONS MADE BY A CLIENT

$$\sigma_{A.Unique\_Ins\_ID=B.Unique\_Ins\_IDANDB.client\_ID=xANDC.policy\_key=B.policy\_key}(\rho_A(\text{transactions}) \times\rho_B(\text{insurances}) \times\rho_c(\text{policies}))$$

- VIEW ALL CLIENTS UNDER A PARTICULAR AGENT

$$\Pi_{client\_name,client\_ph,client\_email}(\sigma_{agent\_ID=x}(\text{clients}))$$

- CHECK FOR OFFER BY A COMPANY (COLLABORATING)

$$\Pi_{A.discount_factor}(\sigma_{A.company\_ID=B.company\_IDANDA.active=1ANDB.company\_reg\_no=C.company\_reg\_noANDC.client\_ID=x}(\rho_A(\text{offers} \times\rho_B(\text{companies}) \times\rho_c(\text{clients})))$$

AND HAS BEEN WRITTEN AS 'AND' IN THESSE RELATIONAL QUERIES, INSTEAD OF /\ FOR BETTER READABILITY. RELAPLACE 'AND' WITH /\ FOR ALL OF THE ABOVE.

RELATIONAL QUERIES FOR BASIC FEATURES FOR EACH STAKEHOLDER

- VIEW ALL STAFF IN A PARTICULAR BRANCH

$$\Pi_{employee\_name,employee\_ph,employee\_email,employee\_ID,branch\_ID,department,position,salary}(\sigma_{branch_ID=x}(\text{staff}))$$

- VIEW AN AGENT'S PROFILE

$$\Pi_{agent\_name,agent\_ph,agent\_aadhar,commission\_factor}(\sigma_{agent\_ID=x}(\text{agents}))$$

- VIEW ALL CLIENTS THAT CAME THROUGH AN ORGANISATION

$$\Pi_{A.client\_name,A.client\_ph,A.client\_email}(\sigma_{A.company\_reg\_no=B.company\_reg\_noANDB.company\_ID=x}(\rho_A(\text{clients}) \times \rho_B(\text{companies})))$$

- VIEW COLLABORATION DETAILS (ORGANISATION)

$$\Pi_{A.offer\_desc,B.collab\_start_date,B.collab\_duration}(\sigma_{B.company\_ID=xANDA.company\_ID=B.company\_IDANDA.active=1}(\rho_A(\text{offers}) \times \rho_B(\text{companies})))$$

- VIEW A PARTICULAR CLIENT'S PROFILE (BY STAFF)

$$\Pi_{client\_name,client\_ph,client\_email,branch\_ID,client\_sex,agent\_name,agent\_ph,agent\_email}(\sigma_{c.client_ID=xANDc.agent_ID=a.agent_ID}(\rho_A(\text{agents}) \times \rho_c(\text{clients})))$$

AND HAS BEEN WRITTEN AS 'AND' IN THESSE RELATIONAL QUERIES, INSTEAD OF /\ FOR BETTER READABILITY. RELAPLACE 'AND' WITH /\ FOR ALL OF THE ABOVE.

WEEK 7

EMBEDDED SQL QUERIES

CLIENTS

VIEW PROFILE

```
@client.context_processor
def viewprofile():
    if not(userLoggedIn() and userType('client')):
        return
    dbCursor = db.cursor()
    sql = "SELECT A.client_ph,A.client_name, A.client_aadhar, A.client_PAN,A.client_DOB, A.client_sex \
        ,B.agent_name, B.agent_ph, B.agent_email, C.branch_name, A.client_ID FROM clients A, agents B, branch C WHERE A.client_ID=%s AND A.agent_ID = B.agent_ID AND A.branch_ID = C.branch_ID"
    val = (session['id'],)
    dbCursor.execute(sql, val)
    res = dbCursor.fetchone()
    dbCursor.close()
    ret = [session['username'], res[0], session['email']] + list(res)[1:]
    return {'clientInfo' : ret }
```

VIEW THE POLICIES AVAILABLE

```
@client.context_processor
def viewBuyPolicies():
    if not(userLoggedIn() and userType('client')):
        return
    dbCursor = db.cursor()
    sql = "SELECT ins_type,policy_name,policy_key, coverage_amt, premium, duration, eligibility_cond,\
    terms_conditions FROM policies"
    dbCursor.execute(sql)
    res = dbCursor.fetchall()
    dbCursor.close()
    return {'buyPolicies' : res}
```

VIEW ALL PREVIOUS TRANSACTIONS

```
@client.context_processor
def viewallTransactions():
    if not(userLoggedIn() and userType('client')):
        return
    dbCursor = db.cursor()
    sql = "SELECT A.payment_datetime, A.amount, B.Unique_Ins_ID, C.ins_type, C.policy_name \
    FROM transactions A, insurances B, policies C \
    WHERE A.Unique_Ins_ID = B.Unique_Ins_ID AND B.client_ID = %s AND C.policy_key = B.policy_key \
    ORDER BY A.payment_datetime DESC"
    val = (session['id'], )
    dbCursor.execute(sql,val)
    res = dbCursor.fetchall()
    dbCursor.close()
    return {'allTransactions' : res}
```

EMBEDDED SQL QUERIES

CLIENTS

VIEW ALL POLICIES

```
@client.context_processor
def viewallpolicies():
    if not(userLoggedIn() and userType('client')):
        return
    dbCursor = db.cursor()
    sql = "SELECT policy_name, ins_type, coverage_amt, premium,eligibility_cond,\
    terms_conditions FROM policies"
    dbCursor.execute(sql)
    res = dbCursor.fetchall()
    dbCursor.close()
    return {'allPolicies' : res}
```

VIEW DUES

```
@client.context_processor
def getDues():
    if not(userLoggedIn() and userType('client')):
        return
    dbCursor = db.cursor()
    sql = "SELECT B.policy_name, B.coverage_amt ,B.premium, A.dues, A.Unique_ins_ID \
        FROM insurances A, policies B WHERE A.client_ID = %s AND A.policy_key = B.policy_key AND A.dues > 0"
    val = (session['id'],)
    dbCursor.execute(sql, val)
    res = dbCursor.fetchall()
    dbCursor.close()
    return {'dues' : res}
```

PAY DUES FOR A POLICY

```
@client.route("/paydue", methods =['POST'])
def paydue():
    if not(userLoggedIn() and userType('client')):
        return
    dbCursor = db.cursor()
    currDateTime = datetime.today().strftime('%Y-%m-%d %H:%M:%S')
    sql = "SELECT dues FROM insurances WHERE Unique_Ins_ID = %s"
    val = (request.form['id'],)
    dbCursor.execute(sql,val)
    res = dbCursor.fetchone()
    if res:
        amount = res[0]
        sql = "INSERT INTO transactions (transaction_ID, Unique_ins_ID, amount, payment_datetime) VALUES (%s, %s, %s, %s)"
        val = (generateUID(16),request.form['id'],amount,currDateTime)
        dbCursor.execute(sql, val)
        db.commit()
        sql = "UPDATE insurances SET dues=0 WHERE Unique_Ins_ID = %s"
        val = (request.form['id'], )
        dbCursor.execute(sql,val)
        db.commit()
        dbCursor.close()
        flash('Transaction Successful')
        return redirect(url_for('client.dashboardPayDues'))
```

EMBEDDED SQL QUERIES

CLIENTS

VIEW DETAILS OF INSURANCES PURCHASED

```
@client.context_processor
def viewinsurances():
    if not(userLoggedIn() and userType('client')):
        return
    dbCursor = db.cursor()
    sql = "SELECT B.policy_name, B.ins_type, B.coverage_amt, B.premium, A.start_date, B.duration, A.dues \
        FROM insurances A, policies B WHERE A.client_ID = %s AND A.policy_key = B.policy_key"
    val = (session['id'],)
    dbCursor.execute(sql, val)
    res = dbCursor.fetchall()
    dbCursor.close()
    return {'allInsurances' : res}
```

VIEW TOTAL INSURANCES PURCHASED

```
@client.context_processor
def totalInsurances():
    if not(userLoggedIn() and userType('client')):
        return
    dbCursor = db.cursor()
    sql = "SELECT COUNT(*) FROM insurances WHERE client_ID = %s";
    val = (session['id'], )
    dbCursor.execute(sql,val)
    res = dbCursor.fetchone()
    dbCursor.close()
    return {'totalInsurances' : res}
```

CHECK IF ANY OFFERS EXISTS (BONUS)

```
@client.context_processor
def offers():
    if not(userLoggedIn() and userType('client')):
        return
    dbCursor = db.cursor()
    sql = "SELECT A.discount_factor FROM offers A, companies B, clients C \
        WHERE A.company_ID = B.company_ID AND A.active = 1 AND B.company_reg_no = C.company_reg_no AND C.client_ID = %s"
    val = (session['id'], )
    dbCursor.execute(sql, val)
    res = dbCursor.fetchone()
    if res:
        return {'offerValid' : True, 'offerDiscount' : res[0]}
    else:
        return {'offerValid' : False}
```

EMBEDDED QUERIES

CLIENTS

BUY INSURANCE (SELECT WHICH POLICY)

```python
@client.route('/buyInsurance', methods=['POST'])
def boughtInsurance():
    if not(userLoggedIn() and userType('client')):
        return
    dbCursor = db.cursor()
    policy_key = request.form['policy_key']
    sql = "SELECT ins_type FROM policies WHERE policy_key=%s"
    val = (policy_key,)
    dbCursor.execute(sql, val)
    res = dbCursor.fetchone()
    dbCursor.close()
    if not res:
        return
    insType = res[0]
    if insType == "Home":
        return buyHome(request)
    elif insType == "Vehicle":
        return buyVehicle(request)
    elif insType == "Medical":
        return buyMedical(request)
    elif insType == "Travel":
        return buyTravel(request)
    else:
        return buyLife(request)
```

BUY TRAVEL INSURANCE

```python
def buyTravel(req):
    dbCursor = db.cursor()

    unique_ins_id = generateUID(12)
    path = unique_ins_id
    currDate = datetime.today().strftime('%Y-%m-%d')

    sql = "SELECT branch_ID from clients where client_ID = %s"
    val = (session['id'], )
    dbCursor.execute(sql, val)
    branchID = dbCursor.fetchone()[0]

    sql = "INSERT INTO insurances (policy_key, client_ID, start_date, branch_ID, Unique_Ins_ID, dues) values (%s, %s, %s, %s, %s, %s)"
    val = (req.form['policy_key'], session['id'], currDate, branchID, unique_ins_id, 0)
    dbCursor.execute(sql,val)
    db.commit()

    sql = "INSERT INTO travel_insurance (travel_type,travel_details,Unique_Ins_ID,travel_date) VALUES (%s, %s, %s, %s)"
    val = (req.form['travelType'], req.form['details'], unique_ins_id, req.form['date'])

    dbCursor.execute(sql,val)
    db.commit()
    dbCursor.close()

    flash('Purchase Successfull')
    return redirect(url_for('client.dashboardBuy'))
```

EMBEDDED SQL QUERIES

CLIENTS

BUY HOME INSURANCE

```
def buyHome(req):
    if 'file' not in req.files:
        flash('No file Provided')
        return redirect(url_for('client.dashboardBuy'))
    file = req.files['file']

    if file.filename == '':
        flash('No selected file')
        return redirect(url_for('client.dashboardBuy'))
    if file:

        dbCursor = db.cursor()

        unique_ins_id = generateUID(12)
        path = unique_ins_id
        currDate = datetime.today().strftime('%Y-%m-%d')

        sql = "SELECT branch_ID from clients where client_ID = %s"
        val = (session['id'], )
        dbCursor.execute(sql, val)
        branchID = dbCursor.fetchone()[0]

        dbCursor.close()

        dbCursor = db.cursor()

        sql = "INSERT INTO insurances (policy_key, client_ID, start_date, branch_ID, Unique_Ins_ID, dues) values (%s, %s, %s, %s, %s, 0)"
        val = (req.form['policy_key'], session['id'], currDate, branchID, unique_ins_id)
        dbCursor.execute(sql,val)
        db.commit()

        dbCursor.close()

        dbCursor = db.cursor()

        filename = secure_filename(file.filename)
        file.save(path)
        sql = "INSERT INTO home_insurance (Unique_Ins_ID,path_to_prop_papers,prop_location,owners_names,prop_area) VALUES (%s, %s, %s, %s, %s)"
        val =  (unique_ins_id, path, req.form['location'],req.form['ownerName'],int(req.form['area']))
        dbCursor.execute(sql, val)
        db.commit()
        dbCursor.close()
        flash('Purchase Successfull')
        return redirect(url_for('client.dashboardBuy'))

    flash('No selected file')
    return redirect(url_for('client.dashboardBuy'))
```

EMBEDDED SQL QUERIES

CLIENTS

BUY LIFE INSURANCE

```
def buyLife(req):
    dbCursor = db.cursor()

    if 'file' not in req.files:
        flash('No file Provided')
        return redirect(url_for('client.dashboardBuy'))
    file = req.files['file']
    if file.filename == '':
        flash('No selected file')
        return redirect(url_for('client.dashboardBuy'))

    if file:

        dbCursor = db.cursor()

        unique_ins_id = generateUID(12)
        path = unique_ins_id
        currDate = datetime.today().strftime('%Y-%m-%d')

        sql = "SELECT branch_ID from clients where client_ID = %s"
        val = (session['id'], )
        dbCursor.execute(sql, val)
        branchID = dbCursor.fetchone()[0]

        sql = "INSERT INTO insurances (policy_key, client_ID, start_date, branch_ID, Unique_Ins_ID, dues) values (%s, %s, %s, %s, %s, %s)"
        val = (req.form['policy_key'], session['id'], currDate, branchID, unique_ins_id, 0)
        dbCursor.execute(sql,val)
        db.commit()

        filename = secure_filename(file.filename)
        file.save(path)

        sql = "INSERT INTO life_insurance (nominee_name2,nominee_name1,path_to_birth_certificate,Unique_Ins_ID) VALUES (%s, %s, %s, %s)"
        val = (req.form['nom1name'], req.form['nom2name'],path, unique_ins_id,)
        dbCursor.execute(sql, val)
        db.commit()
        dbCursor.close()

        flash('Purchase Successfull')
        return redirect(url_for('client.dashboardBuy'))

    flash('No selected file')
    return redirect(url_for('client.dashboardBuy'))
```

BUY MEDICAL INSURANCE

```
def buyMedical(req):

    if 'file' not in req.files:
        flash('No file Provided')
        return redirect(url_for('client.dashboardBuy'))
    file = req.files['file']
    if file.filename == '':
        flash('No selected file')
        return redirect(url_for('client.dashboardBuy'))

    if file:

        dbCursor = db.cursor()

        unique_ins_id = generateUID(12)
        path = unique_ins_id
        currDate = datetime.today().strftime('%Y-%m-%d')

        sql = "SELECT branch_ID from clients where client_ID = %s"
        val = (session['id'], )
        dbCursor.execute(sql, val)
        branchID = dbCursor.fetchone()[0]

        sql = "INSERT INTO insurances (policy_key, client_ID, start_date, branch_ID, Unique_Ins_ID, dues) values (%s, %s, %s, %s, %s, %s)"
        val = (req.form['policy_key'], session['id'], currDate, branchID, unique_ins_id, 0)
        dbCursor.execute(sql,val)
        db.commit()

        filename = secure_filename(file.filename)
        file.save(path)
        sql = "INSERT INTO medical_insurance (Unique_Ins_ID,path_to_medical_bills,medical_history) VALUES (%s, %s, %s)"
        val =  (unique_ins_id, path, req.form['history'])
        dbCursor.execute(sql, val)
        db.commit()
        dbCursor.close()

        flash('Purchase Successfull')
        return redirect(url_for('client.dashboardBuy'))

    flash('No selected file')
    return redirect(url_for('client.dashboardBuy'))
```

EMBEDDED SQL QUERIES

CLIENTS

BUY VEHICLE INSURANCE

```
def buyVehicle(req):
    dbCursor = db.cursor()


    sql = "SELECT * FROM vehicle_insurance WHERE RC_num = %s"
    val = (req.form['rcno'], )
    dbCursor.execute(sql, val)
    res = dbCursor.fetchone()

    if res:
        flash('RC already linked to another insurance')
        return redirect(url_for('client.dashboardBuy'))



    if 'file' not in req.files:
        flash('No file Provided')
        return redirect(url_for('client.dashboardBuy'))
    file = req.files['file']
    if file.filename == '':
        flash('No selected file')
        return redirect(url_for('client.dashboardBuy'))

    if file:
        unique_ins_id = generateUID(12)
        path = unique_ins_id
        currDate = datetime.today().strftime('%Y-%m-%d')

        sql = "SELECT branch_ID from clients where client_ID = %s"
        val = (session['id'], )
        dbCursor.execute(sql, val)
        branchID = dbCursor.fetchone()[0]

        sql = "INSERT INTO insurances (policy_key, client_ID, start_date, branch_ID, Unique_Ins_ID, dues) values (%s, %s, %s, %s, %s, %s)"
        val = (req.form['policy_key'], session['id'], currDate, branchID, unique_ins_id, 0)
        dbCursor.execute(sql,val)
        db.commit()


        filename = secure_filename(file.filename)
        file.save(path)
        sql = "INSERT INTO vehicle_insurance (Unique_Ins_ID, vehicle_ID, vehicle_type, RC_num, Path_to_RC)" \
            "VALUES (%s, %s, %s, %s, %s)"
        val = (unique_ins_id, req.form['vehicleID'], req.form['type'],req.form['rcno'],path)
        dbCursor.execute(sql, val)
        db.commit()
        dbCursor.close()
        flash('Purchase Successfull')
        return redirect(url_for('client.dashboardBuy'))

    flash('No selected file')
    return redirect(url_for('client.dashboardBuy'))
```

EMBEDDED SQL QUERIES

AGENTS

VIEW PROFILE

```
@agent.context_processor
def viewagentprofile():
    if not(userLoggedIn() and userType('agent')):
        return
    dbCursor = db.cursor()
    sql = "SELECT agent_name,agent_ph,agent_aadhar,commission_factor FROM agents WHERE agent_ID=%s"
    val = (session['id'],)
    dbCursor.execute(sql, val)
    res = dbCursor.fetchone()
    dbCursor.close()
    return {'agentProfile' : [session['username'], res[1], session['email'], res[0], res[2], res[3]]}
```

VIEW SOLD POLICIES

```
@agent.context_processor
def viewsold():
    if not(userLoggedIn() and userType('agent')):
        return
    dbCursor = db.cursor()
    sql = "SELECT B.client_name, B.client_ph, B.client_email ,C.ins_type, A.start_date, C.duration FROM insurances A, clients B, policies C WHERE C.policy_key=A.policy_key AND B.agent_ID=%s AND A.client_ID=B.client_ID"
    val = (session['id'],)
    dbCursor.execute(sql, val)
    res = dbCursor.fetchall()
    dbCursor.close()
    return {'agentPoliciesSold' : res}
```

GET CLIENT CONTACT

```
@agent.context_processor
def getClientContact():
    if not(userLoggedIn() and userType('agent')):
        return
    dbCursor = db.cursor()
    sql = "SELECT client_name, client_ph, client_email \
    FROM clients WHERE agent_ID= %s"
    agent_ID = session['id']
    val = (agent_ID,)
    dbCursor.execute(sql, val)
    res= dbCursor.fetchall()
    dbCursor.close()
    return {'agentClientContact' : res}
```

EMBEDDED SQL QUERIES

AGENTS

GET NUMBER OF CLIENTS

```
@agent.context_processor
def getClientCount():
    if not(userLoggedIn() and userType('agent')):
        return
    dbCursor = db.cursor()
    sql = "SELECT COUNT(*) \
    FROM clients WHERE agent_ID= %s"
    agent_ID = session['id']
    val = (agent_ID, )
    dbCursor.execute(sql, val)
    res= dbCursor.fetchone()[0]
    dbCursor.close()
    return {'agentClientCount' : res}
```

COUNT THE NUMBER OF POLICIES SOLD

```
@agent.context_processor
def viewCountSold():
    if not(userLoggedIn() and userType('agent')):
        return
    dbCursor = db.cursor()
    sql = "SELECT COUNT(*) FROM insurances I, (SELECT client_ID from clients WHERE agent_ID = %s) A WHERE I.client_ID = A.client_ID"
    val = (session['id'],)
    dbCursor.execute(sql, val)
    res = dbCursor.fetchone()[0]
    dbCursor.close()
    return {'agentCountSold' : res}
```

EMBEDDED SQL QUERIES

ADMINS

ADD AGENT

```
@admin.route('/addAgent', methods= ['POST'])
def addAgent():
    if not(userLoggedIn() and userType('admin')):
        return
    dbCursor = db.cursor()
    if validateAddAgentRequest(request.form):
        addUser(request.form, "agent")
        sql = "INSERT INTO agents(agent_name,\
        agent_ph, agent_email, agent_aadhar, agent_ID, commission_factor) VALUES (%s, %s, %s, %s, %s, %s)"
        val = (request.form['name'], request.form['phone'],
        request.form['email'], request.form['aadhar'],
        generateUID(12), request.form['commission'])
        dbCursor.execute(sql, val)
        db.commit()
        dbCursor.close()
    flash("Successfully Registered!","success")
    return redirect(url_for('admin.dashboardAddAgent'))
```

ADD STAFF

```
@admin.route('/addStaff', methods= ['POST'])
def addStaff():
    if not(userLoggedIn() and userType('admin')):
        return
    dbCursor = db.cursor()
    if validateAddStaffRequest(request.form):
        addUser(request.form, "employee")
        sql = "INSERT INTO staff(\
        employee_name,employee_ph,employee_email,employee_aadhar,employee_PAN,\
        employee_ID,branch_ID, department, position, salary) VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s, %s)"

        val = (request.form['name'], request.form['phone'],
        request.form['email'], request.form['aadhar'],
        request.form['pan'], generateUID(12),
        request.form['branch'], request.form['dept'],
        request.form['pos'], request.form['salary'])
        dbCursor.execute(sql, val)
        db.commit()
        dbCursor.close()
    flash("Successfully Registered!","success")
    return redirect(url_for('admin.dashboardAddEmp'))
```

EMBEDDED SQL QUERIES

ADMINS

CHECK PROFIT OF BRANCH

```
@admin.context_processor
def checkProfit():
    if not(userLoggedIn() and userType('admin')):
        return
    dbCursor = db.cursor()
    sql = "SELECT   A.branch_ID,   B.branch_name,   SUM(A.P) AS profit FROM   (SELECT i.branch_ID,\
          SUM(p.premium - p.coverage_amt) AS P     FROM insurances i, policies p WHERE\
                p.policy_key = i.policy_key AND DATE_ADD(start_date, INTERVAL duration YEAR) <= curdate()\
               GROUP BY branch_ID UNION ALL SELECT  i.branch_ID, SUM(TIMESTAMPDIFF(MONTH,\
                          i.start_date,curdate()) * p.ppm) AS P FROM \
                          insurances i, policies p     WHERE \
                          DATE_ADD(start_date, INTERVAL duration YEAR) > curdate()     GROUP BY       i.branch_ID     \
                          UNION ALL SELECT branch.branch_ID, 0 AS P FROM branch) AS A   \
                          INNER JOIN \
    branch B ON A.branch_ID = B.branch_ID GROUP BY   B.branch_ID"
    dbCursor.execute(sql)
    res = dbCursor.fetchall()
    dbCursor.close()
    print(res)
    return {'branchProfit' : res}
```

REMOVE LOGINS (USER) FROM DATABASE

```
@admin.route('/removeLogin', methods=['POST'])
def remLogin():
    if not(userLoggedIn() and userType('admin')):
        return
    dbCursor = db.cursor()
    email = request.form['ID']
    sql = "DELETE FROM login WHERE email=%s"
    val=(email, )
    dbCursor.execute(sql,val)
    db.commit()
    dbCursor.close()
    return redirect(url_for('admin.dashboardDeactivateAcc'))
```

EMBEDDED SQL QUERIES

ADMINS

VIEW BRANCH DETAILS

```
@admin.app_context_processor
def viewBranchDetails():
    # if not(userLoggedIn() and (userType('admin') or userType('shareholders'))):
    #     return
    dbCursor = db.cursor()
    sql = "SELECT branch_name, branch_ph, branch_email, \
    (select (select count(*) from staff s where s.branch_ID=b.branch_ID) from dual) as total_employees\
    , (select (select count(*) from insurances i where i.branch_ID=b.branch_ID) from dual) \
    as policies_sold, branch_ID from branch b;"
    dbCursor.execute(sql)
    res = dbCursor.fetchall()
    dbCursor.close()
    return {'branchDetails' :  res}
```

VIEW STAFF IN A PARTICULAR BRANCH

```
@admin.route('/viewbranchStaff', methods= ['POST'])
def viewbranchStaff():
    if not(userLoggedIn() and userType('admin')):
        return
    dbCursor = db.cursor()
    branchID = request.values.get('branchID')
    sql = "SELECT employee_name, employee_ph, employee_email,\
    employee_aadhar,employee_PAN,employee_ID, branch_ID, department, position, salary FROM staff WHERE branch_ID = %s "
    val = (branchID, )
    dbCursor.execute(sql, val)
    res = dbCursor.fetchall()
    dbCursor.close()
    if res:
        return render_template('admin/branchEmp.html', branchEmpQuery=True, branchEmp=res)
    else:
        return render_template('admin/branchEmp.html', branchEmpQuery=False)
```

VIEW LIST OF PEOPLE WHO CAN ACCESS THE FIRM

```
@admin.context_processor
def viewLogins():
    if not(userLoggedIn() and userType('admin')):
        return
    dbCursor = db.cursor()
    sql = "SELECT username, email FROM login"
    dbCursor.execute(sql)
    res = dbCursor.fetchall()
    dbCursor.close()
    return {'allLogins' : res}
```

EMBEDDED SQL QUERIES

STAFF

VIEW CLIENT DETAILS

```
@staff.route('/viewClientDetails', methods = ["POST"])
def viewClientDetails():
    if not(userLoggedIn() and userType("employee")):
        return
    dbCursor = db.cursor()
    sql = "SELECT client_name,client_ph,client_email,branch_ID,client_aadhar,client_PAN, "\
    "client_DOB,client_sex,agent_name,agent_ph,agent_email  FROM clients c, agents a WHERE "\
    "c.client_ID = %s AND c.agent_ID=a.agent_ID"
    val = (request.form['clientID'],)
    dbCursor.execute(sql, val)
    res = dbCursor.fetchone()
    sql = "SELECT Unique_Ins_ID, ins_type, DATE_ADD(start_date, INTERVAL duration YEAR) as end_date  FROM \
    insurances, policies WHERE policies.policy_key=insurances.policy_key AND client_ID = %s"
    val = (request.form['clientID'],)
    dbCursor.execute(sql, val)
    res2 = dbCursor.fetchall()
    dbCursor.close()
    if res:
        return render_template('staff/clientInfo.html',staffClientQuery=True, clientInfo=res, staffClientIns=res2)
    else:
        return render_template('staff/clientInfo.html',staffClientQuery=False)
```

VIEW DETAILS OF POLICIES SOLD

```
@staff.route("/viewStaffInsurance", methods = ['POST'])
def viewInsurance():
    if not(userLoggedIn() and userType('employee')):
        return
    dbCursor = db.cursor()
    sql = "SELECT c.client_name, c.client_ID,c.agent_ID,ins_type, " \
    "policy_name,coverage_amt, ppm, start_date,  DATE_ADD(start_date, INTERVAL duration YEAR), "\
    "dues, Unique_Ins_id FROM clients c, insurances i, policies p WHERE "\
    "i.Unique_Ins_id = %s AND c.client_ID=i.client_ID AND p.policy_key=i.policy_key"
    val = (request.form['insID'],)
    dbCursor.execute(sql, val)
    res = dbCursor.fetchone()
    dbCursor.close()
    if res:
        return render_template('staff/insurance.html',staffInsQuery=True, insInfo=res)
    else:
        return render_template('staff/insurance.html',staffInsQuery=False)
```

VIEW PROFILE

```
@staff.context_processor
def viewStaffProfile():
    if not(userLoggedIn() and userType('employee')):
        return
    dbCursor = db.cursor()
    sql = "SELECT A.employee_name, \
    A.employee_aadhar, A.employee_PAN, B.branch_name,A.department, A.position, A.salary, A.employee_ph \
    FROM staff A, branch B  \
    WHERE employee_ID = %s AND A.branch_ID=B.branch_ID"
    employee_ID = session['id']
    val = (employee_ID,)
    dbCursor.execute(sql, val)
    res = dbCursor.fetchone()
    dbCursor.close()
    return {'staffProfile' : [session['username'], res[7], session['email'], res[0], res[1], res[2],res[3], res[4],res[5],res[6]]}
```

EMBEDDED SQL QUERIES

SHAREHOLDERS

VIEW NUMBER OF ACTIVE INSURANCES

```
@shareholders.app_context_processor
def activeInsuranceCounts():
    dbCursor = db.cursor()
    currDate = datetime.today().strftime('%Y-%m-%d')
    sql = "SELECT COUNT(*) AS count FROM insurances A, policies B WHERE A.policy_key=B.policy_key AND DATE_ADD(A.start_date, INTERVAL B.duration YEAR) > %s"
    val = (currDate, )
    dbCursor.execute(sql, val)
    res = dbCursor.fetchone()
    dbCursor.close()
    return {'activeInsurances' : res}
```

GET ANNUAL PROFIT OF THE FIRM

```
@shareholders.app_context_processor
def getAnnualProfit():
    years = [2015,2016,2017,2018,2019]
    res = []
    for year in years:
        res.append((year,int(annualProfit(year)[0])))
    return {'annualProfit' : res}

def annualProfit(year):
    dbCursor = db.cursor()
    startDate = '%d-01-01' %(year)
    endDate = '%d-12-31' %(year)
    sql = "SELECT SUM(A.P) as profit FROM " \
        "(SELECT SUM(-1*B.coverage_amt) as P FROM insurances A, policies B WHERE A.policy_key = B.policy_key AND DATE_ADD(A.start_date, INTERVAL B.duration YEAR) \
        <= %s AND DATE_ADD(A.start_date, INTERVAL B.duration YEAR) >= %s UNION ALL " \
        "SELECT SUM(TIMESTAMPDIFF(MONTH, %s, DATE_ADD(A.start_date, INTERVAL B.duration YEAR))*B.ppm) as P FROM insurances A, policies B WHERE \
        A.policy_key = B.policy_key AND DATE_ADD(A.start_date, INTERVAL B.duration YEAR) <= %s AND DATE_ADD(A.start_date, INTERVAL B.duration YEAR) >= %s UNION ALL " \
        "SELECT SUM(12*B.ppm) as P FROM insurances A, policies B WHERE A.policy_key = B.policy_key AND A.start_date <= %s AND DATE_ADD(A.start_date, INTERVAL B.duration YEAR) >= %s UNION ALL " \
        "SELECT SUM(TIMESTAMPDIFF(MONTH, A.start_date, %s)*B.ppm) as P FROM insurances A, policies B WHERE A.policy_key = B.policy_key AND  A.start_date >= %s AND A.start_date <= %s AND \
        DATE_ADD(A.start_date, INTERVAL B.duration YEAR) >= %s UNION ALL " \
        "SELECT 0 as P FROM insurances ) AS A "
    val = (endDate, startDate, startDate, endDate, startDate, startDate, endDate, endDate, startDate, endDate, endDate)
    dbCursor.execute(sql, val)
    res = dbCursor.fetchone()
    dbCursor.close()
    return res
```

VIEW NUMBER OF ACTIVE (SPECIFIC) INSURANCE

```
@shareholders.app_context_processor
def howManyactiveXYZ():
    dbCursor = db.cursor()
    currDate = datetime.today().strftime('%Y-%m-%d')
    sql = "SELECT B.ins_type, COUNT(*) AS count FROM insurances A, policies B \
        WHERE A.policy_key=B.policy_key AND DATE_ADD(A.start_date, INTERVAL B.duration YEAR) > %s \
        GROUP BY B.ins_type"
    val = (currDate, )
    dbCursor.execute(sql, val)
    res = dbCursor.fetchall()
    dbCursor.close()
    return {'activeXYZ' : res}
```

GET NET PROFIT

```
@shareholders.app_context_processor
def howManytotalXYZ():
    dbCursor = db.cursor()
    currDate = datetime.today().strftime('%Y-%m-%d')
    sql = "SELECT B.ins_type, COUNT(*) AS count FROM insurances A, policies B WHERE A.policy_key = B.policy_key GROUP BY ins_type"
    dbCursor.execute(sql)
    res = dbCursor.fetchall()
    dbCursor.close()
    return {'totalXYZ' : res}
```

EMBEDDED SQL QUERIES

SHAREHOLDERS

VIEW NUMBER OF TOTAL (SPECIFIC) INSURANCES

```
@shareholders.app_context_processor
def howManytotalXYZ():
    dbCursor = db.cursor()
    currDate = datetime.today().strftime('%Y-%m-%d')
    sql = "SELECT B.ins_type, COUNT(*) AS count FROM insurances A, policies B WHERE A.policy_key = B.policy_key GROUP BY ins_type"
    dbCursor.execute(sql)
    res = dbCursor.fetchall()
    dbCursor.close()
    return {'totalXYZ' : res}
```

GET TOTAL NUMBER OF INSURANCES SOLD

```
@shareholders.app_context_processor
def totalInsuranceCounts():
    dbCursor = db.cursor()
    currDate = datetime.today().strftime('%Y-%m-%d')
    sql = "SELECT COUNT(*) AS count FROM insurances"
    dbCursor.execute(sql)
    res = dbCursor.fetchone()
    dbCursor.close()
    return {'totalInsurances' : res}
```

VIEW PROFILE

```
@shareholders.context_processor
def shareUserProfile():
    dbCursor = db.cursor()
    sql = "SELECT equity_percentage, share_name FROM shareholders WHERE share_ID = %s"
    val = (session['id'], )
    dbCursor.execute(sql, val)
    res = dbCursor.fetchone()
    dbCursor.close()
    return {'shareUserProfile' : [session['username'], session['email'], res[1], res[0]]}
```

EMBEDDED SQL QUERIES

ORGANIZATIONS

VIEW NUMBER OF CLIENTS THROUGH THIS ORG.

```
@organizations.context_processor
def viewOrgNumberClients():
    if not(userLoggedIn() and userType('company')):
        return
    dbCursor = db.cursor()
    sql = "SELECT COUNT(*) FROM clients A, companies B WHERE A.company_reg_no = B.company_reg_no AND B.company_ID = %s "
    val = (session['id'], )
    dbCursor.execute(sql, val)
    res = dbCursor.fetchone()
    dbCursor.close()
    return {'orgClientCount' : res}
```

VIEW COLLAB DETAILS

```
@organizations.context_processor
def viewCollabDetails():
    if not(userLoggedIn() and userType('company')):
        return

    dbCursor = db.cursor()

    sql = "SELECT A.offer_desc, B.collab_start_date, B.collab_duration FROM offers A, companies B WHERE B.company_ID = %s AND A.company_ID = B.company_ID AND A.active = 1"
    val = (session['id'], )
    dbCursor.execute(sql, val)
    res = dbCursor.fetchone()
    dbCursor.close()

    if res:
        return {'collabExists' : True, 'collabDetails' : res}
    else:
        return {'collabExists' : False}
```

VIEW ALL CLIENTS THROUGH THIS ORGANIZATION

```
@organizations.context_processor
def viewOrgClients():
    if not(userLoggedIn() and userType('company')):
        return
    dbCursor = db.cursor()
    sql = "SELECT A.client_name, A.client_ph, A.client_email FROM clients A, companies B WHERE A.company_reg_no = B.company_reg_no AND B.company_ID = %s "
    val = (session['id'], )
    dbCursor.execute(sql, val)
    res = dbCursor.fetchall()
    dbCursor.close()
    return {'orgClients' : res}
```

VIEW PROFILE

```
@organizations.context_processor
def viewOrgProfile():
    if not(userLoggedIn() and userType('company')):
        return
    dbCursor = db.cursor()
    company_ID = session['id']
    sql = "SELECT company_ph, company_name, company_reg_no FROM companies WHERE company_ID = %s "
    val = (company_ID, )
    dbCursor.execute(sql, val)
    res = dbCursor.fetchone()
    dbCursor.close()
    return {'orgProfile' : [session['username'], res[0], session['email'], res[1], res[2]]}
```

EMBEDDED SQL QUERIES

ORGANIZATIONS

APPLY FOR COLLABORATION

```
@organizations.route('/applyCollab', methods= ['POST'])
def applyCollab():
    if not(userLoggedIn() and userType('company')):
        return

    if viewCollabDetails()['collabExists']:
        return

    dbCursor = db.cursor()

    currDate = datetime.today().strftime('%Y-%m-%d')
    offerDesc = None
    discountFactor = None
    if request.form['offer'] == "1":
        offerDesc = "Flat Discount"
        discountFactor = 0.2
    else:
        offerDesc = "Limited Discount"
        discountFactor = 0.3

    sql = "UPDATE companies SET collab_start_date = %s, collab_duration = %d WHERE company_ID = %s"
    val = (currDate, request.form['duration'])
    dbCursor.execute(sql, val)
    db.commit()

    offerID = generateUID(6)
    sql = "INSERT INTO offers (company_ID, offer_desc, discount_factor, offer_ID, active) VALUES (%s, %s, %f, %s, %d)"
    val = (session['id'],offerDesc, discountFactor, offerID, 1)
    dbCursor.execute(sql, val)
    db.commit()

    dbCursor.close()
    return redirect(url_for('organizations.dashboardCollab'))
```

EXTEND COLLABORATION

```
@organizations.route('/extendCollabDuration', methods= ['POST'])
def extendCollabDuration():
    if not(userLoggedIn() and userType('company')):
        return
    dbCursor = db.cursor()
    sql = "UPDATE companies SET collab_duration = collab_duration + " + str(int(request.form['extension'])) + " WHERE company_ID = %s "
    val = (session['id'],)
    dbCursor.execute(sql, val)
    db.commit()
    dbCursor.close()
    return redirect(url_for('organizations.dashboardCollab'))
```

EMBEDDED SQL QUERIES

OTHER GENERAL QUERIES

ADDING A NEW COMPANY (BACK-END)

```
def addCompany(requestForm):
    dbCursor = db.cursor()

    currDate = datetime.today().strftime('%Y-%m-%d')
    offerDesc = None
    discountFactor = None
    if requestForm['offer'] == "1":
        offerDesc = "Flat Discount"
        discountFactor = 0.2
    else:
        offerDesc = "Limited Discount"
        discountFactor = 0.3

    company_ID = generateUID(12)

    sql = "INSERT INTO companies (collab_duration,collab_start_date,company_name,company_ID,company_ph,company_email,company_reg_no) \
        VALUES (%s, %s, %s, %s, %s, %s, %s)"
    val = (requestForm['duration'], currDate,
    requestForm['name'], company_ID,
    requestForm['phone'],requestForm['email'],
    requestForm['regNo'])

    dbCursor.execute(sql, val)
    db.commit()


    offerID = generateUID(6)
    sql = "INSERT INTO offers (company_ID, offer_desc, discount_factor, offer_ID, active) VALUES (%s, %s, %s, %s, %s)"
    val = (company_ID,offerDesc, discountFactor, offerID, 1)
    dbCursor.execute(sql, val)
    db.commit()
```

VALIDATE A LOGIN ATTEMPT

```
def validLogin(email, password):
    dbCursor = db.cursor()
    sql = "SELECT * FROM login WHERE email = %s AND password = %s"
    val = (email, password)
    dbCursor.execute(sql, val)
    res = True if dbCursor.fetchone() else False
    dbCursor.close()
    return res
```

GET AN AGENT'S ID

```
def getAgentID():
    dbCursor = db.cursor(buffered=True)
    sql = "SELECT agent_ID, COUNT(agent_ID) FROM clients GROUP BY agent_ID ORDER BY COUNT(agent_ID) ASC"
    dbCursor.execute(sql)
    agentID = dbCursor.fetchall()[0][0]
    dbCursor.close()
    return agentID
```

EMBEDDED SQL QUERIES

OTHER GENERAL QUERIES

GETTING USER'S INFORMATION

```
def getUserInfo(email):
    dbCursor = db.cursor()
    sql = "SELECT user_type, username FROM login WHERE email = %s"
    val = (email, )
    dbCursor.execute(sql,val)
    res = dbCursor.fetchone()
    usertype = res[0]
    username = res[1]
    userID = 0
    if usertype == 'client':
        sql = "SELECT client_ID FROM clients WHERE client_email = %s"
        val = (email,)
        dbCursor.execute(sql,val)
        userID = dbCursor.fetchone()[0]
    elif usertype == 'employee':
        sql = "SELECT employee_ID FROM staff WHERE employee_email = %s"
        val = (email,)
        dbCursor.execute(sql,val)
        userID = dbCursor.fetchone()[0]
    elif usertype == 'agent':
        sql = "SELECT agent_ID FROM agents WHERE agent_email = %s"
        val = (email,)
        dbCursor.execute(sql,val)
        userID = dbCursor.fetchone()[0]
    elif usertype == 'shareholder':
        sql = "SELECT share_ID FROM shareholders WHERE share_email = %s"
        val = (email,)
        dbCursor.execute(sql,val)
        userID = dbCursor.fetchone()[0]
    elif usertype == 'company':
        sql = "SELECT company_ID FROM companies WHERE company_email = %s"
        val = (email,)
        dbCursor.execute(sql,val)
        userID = dbCursor.fetchone()[0]

    dbCursor.close()
    return userID, username, usertype
```

ADD A NEW CLIENT (BACK-END)

```
def addClient(requestForm):
    dbCursor = db.cursor()
    agentID = getAgentID()
    sql = "INSERT INTO clients (client_PAN,client_DOB,client_name,client_ph,branch_ID,agent_ID,client_email,client_sex,client_ID,company_reg_no,client_aadhar) \
        VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s, NULL, %s)"
    val = (requestForm['pan'], requestForm['dob'],
    requestForm['name'], requestForm['phone'],
    requestForm['branch'], getAgentID(),
    requestForm['email'], getGender(requestForm['sex']),
    generateUID(12),requestForm['aadhar'])
    dbCursor.execute(sql, val)
    db.commit()
    dbCursor.close()
```

EMBEDDED SQL QUERIES

OTHER GENERAL QUERIES

ADDING A NEW USER (BACK-END)

```
def addUser(requestForm, tp=""):
    dbCursor = db.cursor()
    sql = "INSERT INTO login (username, password, email, user_type) VALUES (%s, %s, %s, %s) "
    val = (requestForm['username'], requestForm['password'], requestForm['email'], tp)
    dbCursor.execute(sql, val)
    db.commit()
    dbCursor.close()
    if tp=='client':
        addClient(requestForm)
    if tp=='company':
        addCompany(requestForm)
```

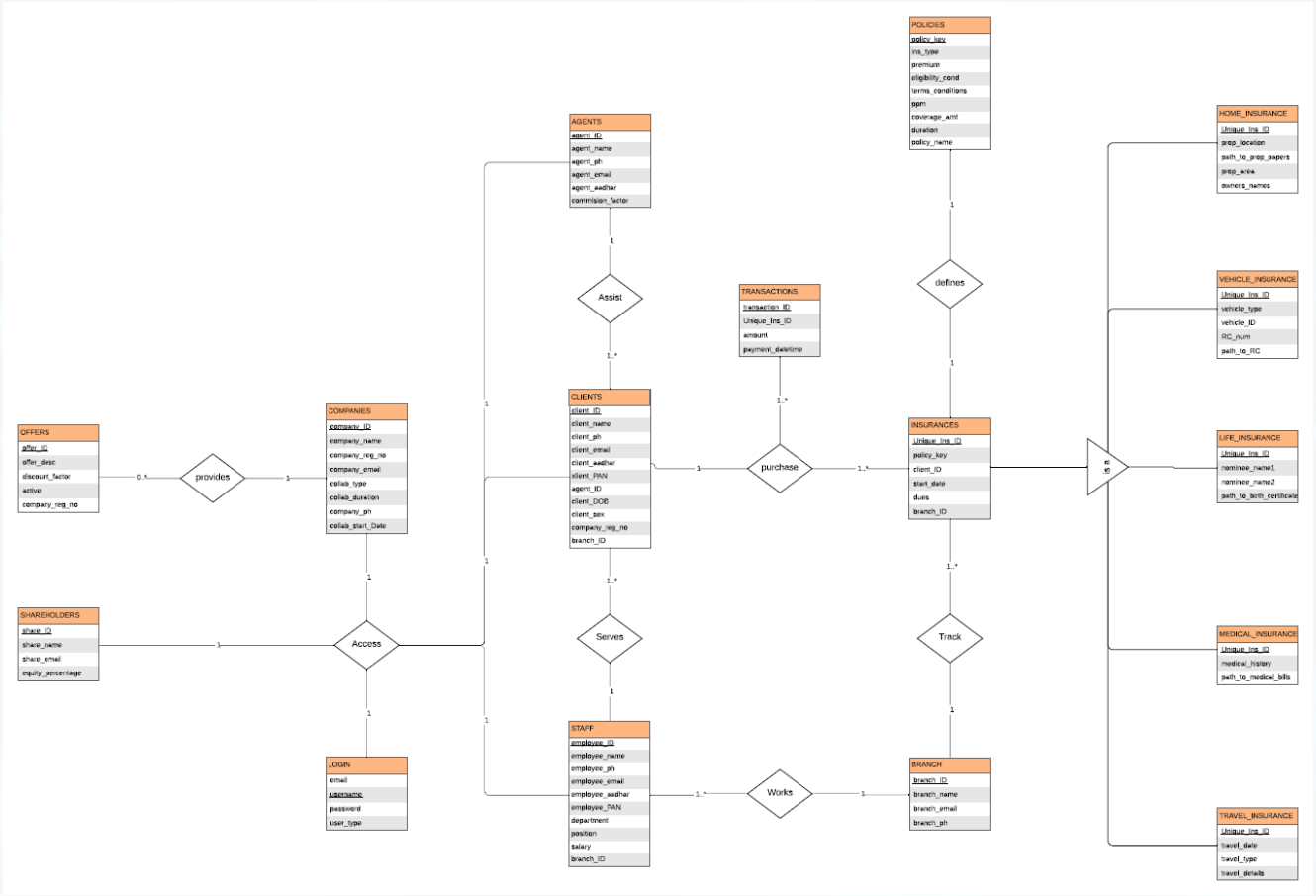
UPDATING DUES AND WEB APP AT MIDNIGHT (BONUS)

```
app = None

def updateDues():
  currDate = date.today()
  if currDate > app.config['currDate']:
    dbCursor = db.cursor()
    sql = "UPDATE (insurances A INNER JOIN policies B ON A.policy_key = B.policy_key) \
      INNER JOIN clients C on A.client_ID = C.client_ID \
      SET A.dues=A.dues+B.ppm WHERE TIMESTAMPDIFF(MONTH, A.start_date, curdate()) > \
      TIMESTAMPDIFF(MONTH, A.start_date, DATE_SUB(curdate(), INTERVAL 1 DAY)) AND \
      DATE_ADD(start_date, INTERVAL B.duration YEAR) >= curdate() AND C.company_reg_no IS NULL;"
    dbCursor.execute(sql)
    db.commit()
    sql = "UPDATE (((insurances A INNER JOIN policies B ON A.policy_key = B.policy_key) \
      INNER JOIN clients C on A.client_ID = C.client_ID) \
      INNER JOIN companies D ON D.company_reg_no = C.company_reg_no) \
      INNER JOIN offers E on D.company_ID = E.company_ID \
      SET A.dues=A.dues+(B.ppm*(100-E.discount_factor)/100) \
      WHERE TIMESTAMPDIFF(MONTH, A.start_date, curdate()) > \
      TIMESTAMPDIFF(MONTH, A.start_date, DATE_SUB(curdate(), INTERVAL 1 DAY)) \
      AND DATE_ADD(start_date, INTERVAL B.duration YEAR) >= curdate() AND E.active=1;"
    dbCursor.execute(sql)
    db.commit()
    dbCursor.close()
  app.config['currDate'] = currDate
```

WEEK 8

E-R DIAGRAM

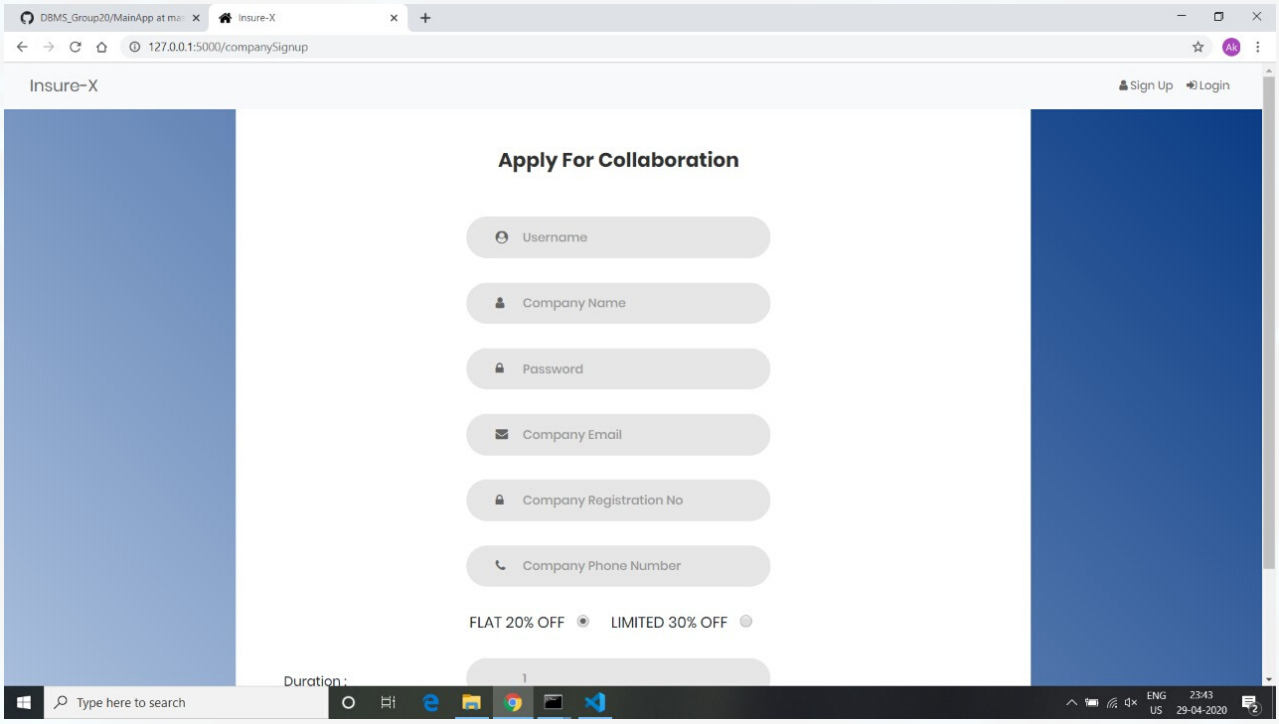


(Click on the image to view full-size image of the E-R Diagram)

WEEK 9

INNOVATIVE FEATURE 1: COLLABORATION

COLLABORATION BETWEEN INSURE-X AND OTHER COMPANIES



Collaborating with other organizations, like travel agencies, airlines or railways, banks etc, will bring clients to the firm.

For example, a client can purchase travel insurance through an airline who is our collaborating partner. This insurance is sold by the airlines, but is essentially comes under our firm.

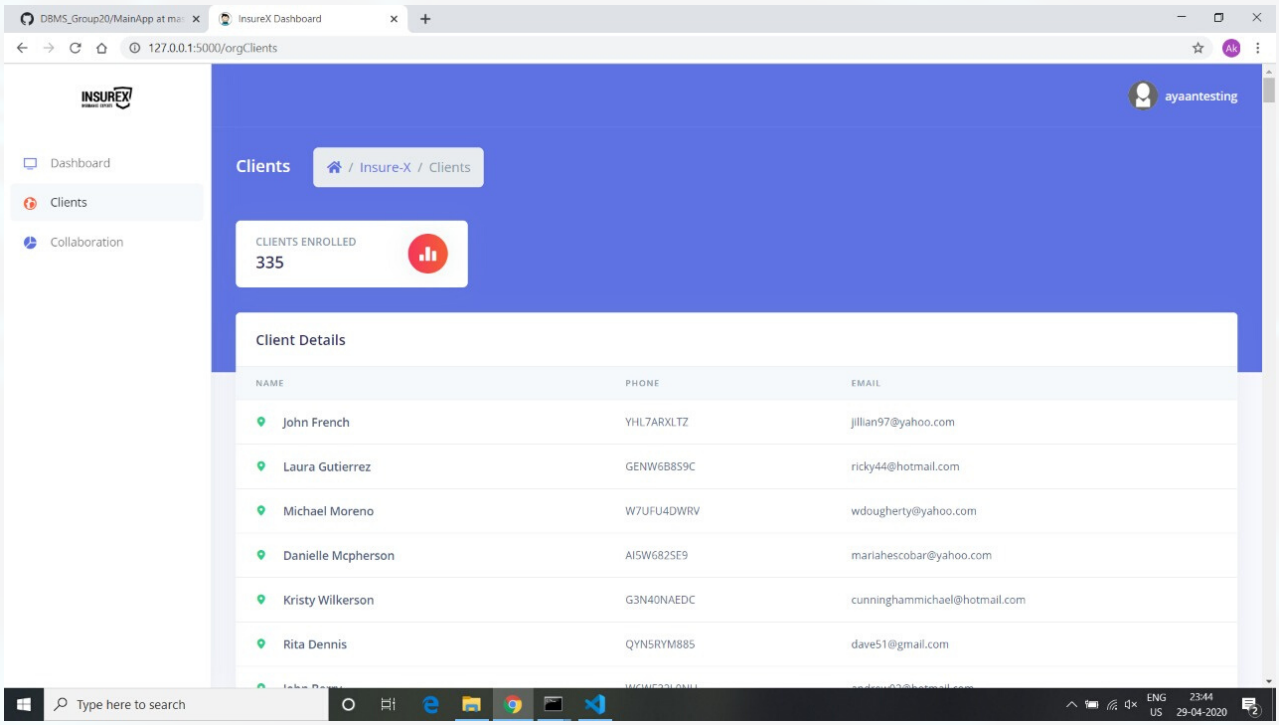
Similarly, banks, as usual, can sell our insurances when a person gets an account opened in that bank. These clients come through the bank, but hold an insurance by Insure-X.

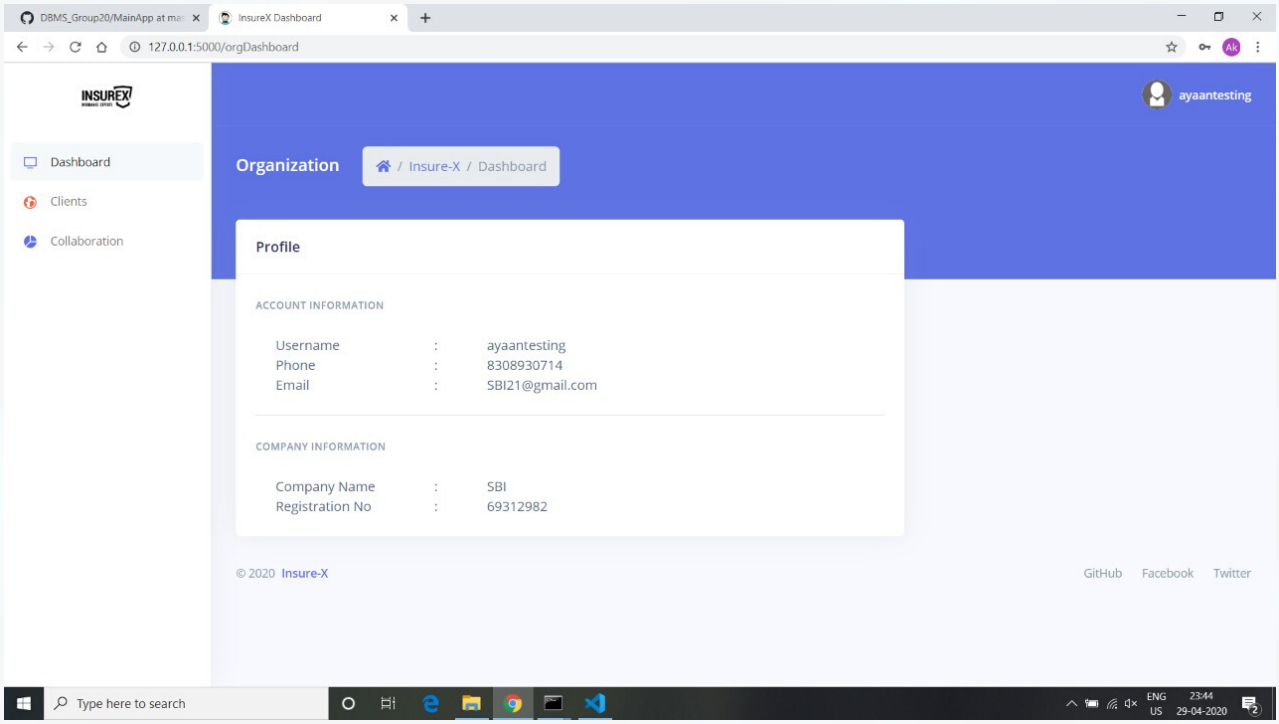
Collaboration can be for a particular duration, which can be specified while applying for collaboration. This duration can also be extended.

The collaborating company can then log onto our portal and view all details about the collaboration, as well as the clients that have come through it to our firm. Screenshots have been attached below.

INNOVATIVE FEATURE 1: COLLABORATION

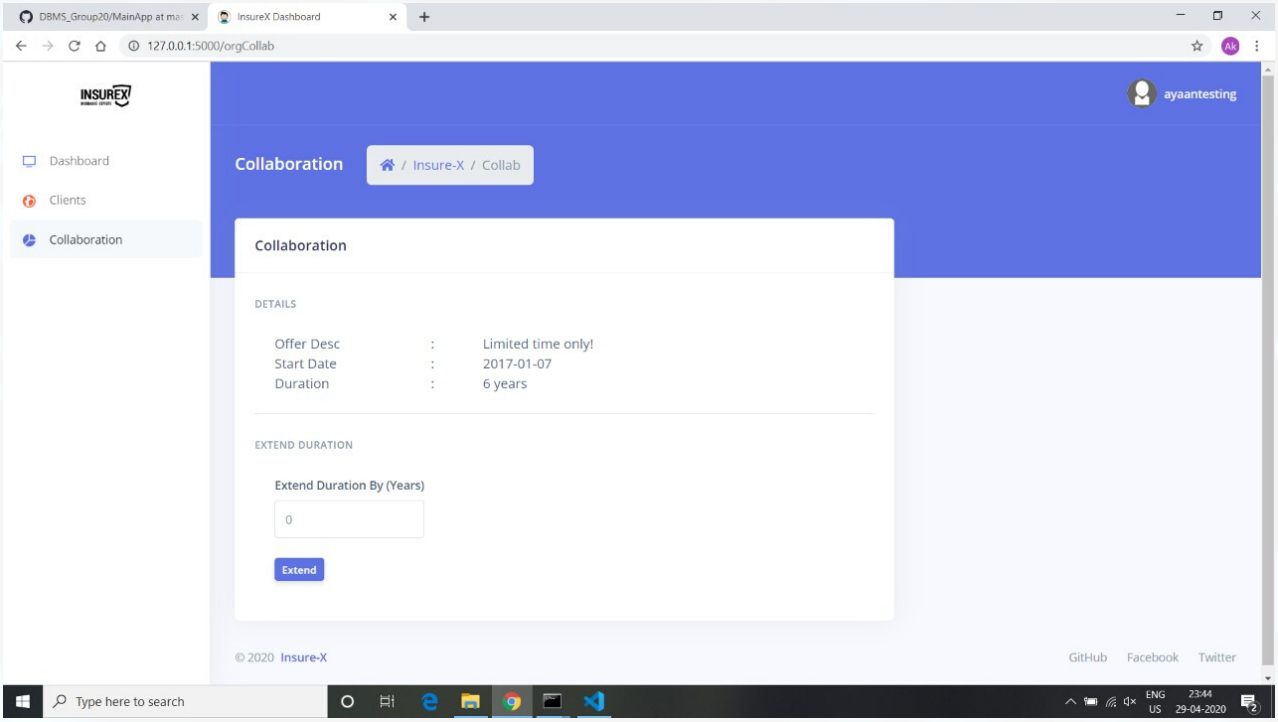
COLLABORATION BETWEEN INSURE-X AND OTHER COMPANIES





INNOVATIVE FEATURE 2: OFFERS

COMPANIES CAN OFFER DISCOUNTS TO THE CLIENTS ENROLLED THROUGH THEM



Collaborating companies, can select a discount package that they can offer to clients that have enrolled through them.

This discount reflects in the "Dues" that the client has to pay for that month. The dues (ppm) are reduced by the discount factor that the company selects.

The offer is applied automatically to all clients through the collaborating company.

INNOVATIVE FEATURE 3: MIDNIGHT UPDATE

MIDNIGHT UPDATE OF THE DUES, AND COMPLETE WEB APPLICATION EVERYDAY

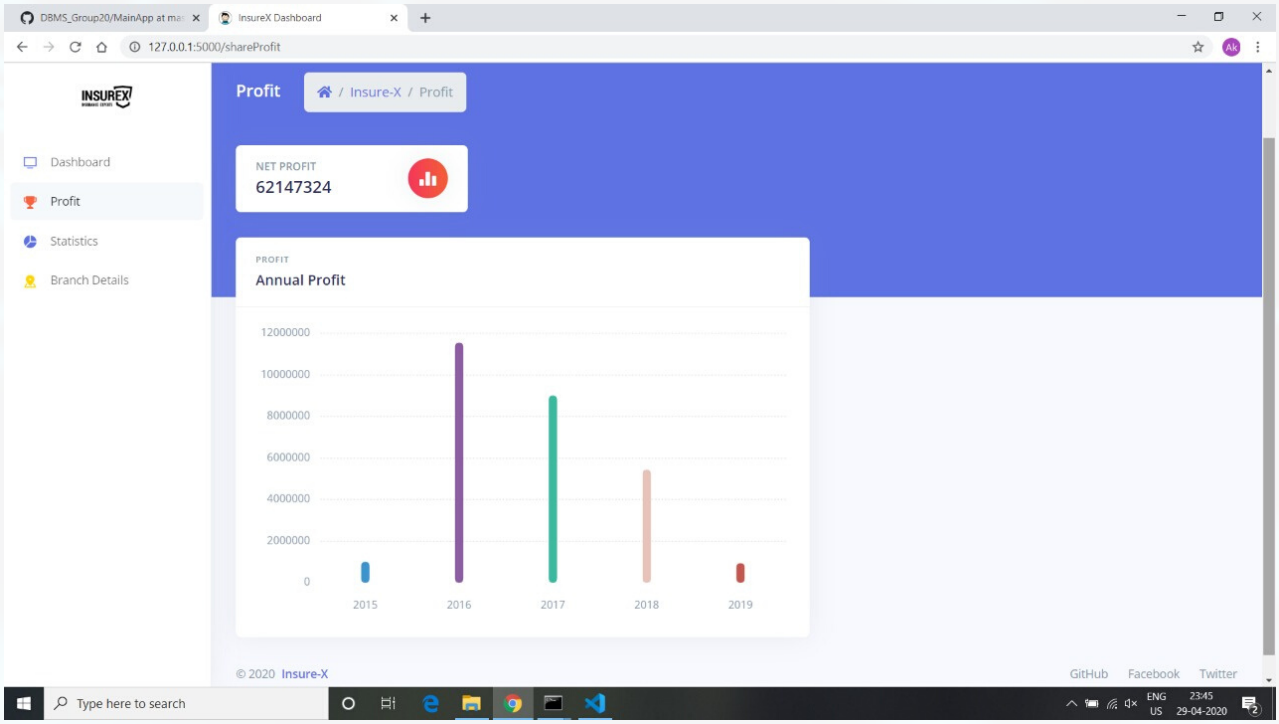
```
app = None

def updateDues():
  currDate = date.today()
  if currDate > app.config['currDate']:
    dbCursor = db.cursor()
    sql = "UPDATE (insurances A INNER JOIN policies B ON A.policy_key = B.policy_key) \
      INNER JOIN clients C on A.client_ID = C.client_ID \
      SET A.dues=A.dues+B.ppm WHERE TIMESTAMPDIFF(MONTH, A.start_date, curdate()) > \
      TIMESTAMPDIFF(MONTH, A.start_date, DATE_SUB(curdate(), INTERVAL 1 DAY)) AND \
      DATE_ADD(start_date, INTERVAL B.duration YEAR) >= curdate() AND C.company_reg_no IS NULL;"
    dbCursor.execute(sql)
    db.commit()
    sql = "UPDATE (((insurances A INNER JOIN policies B ON A.policy_key = B.policy_key) \
      INNER JOIN clients C on A.client_ID = C.client_ID) \
      INNER JOIN companies D ON D.company_reg_no = C.company_reg_no) \
      INNER JOIN offers E on D.company_ID = E.company_ID \
      SET A.dues=A.dues+(B.ppm*(100-E.discount_factor)/100) \
      WHERE TIMESTAMPDIFF(MONTH, A.start_date, curdate()) > \
      TIMESTAMPDIFF(MONTH, A.start_date, DATE_SUB(curdate(), INTERVAL 1 DAY)) \
      AND DATE_ADD(start_date, INTERVAL B.duration YEAR) >= curdate() AND E.active=1;"
    dbCursor.execute(sql)
    db.commit()
    dbCursor.close()
  app.config['currDate'] = currDate
```

This code refreshes the entire web application and updates the dues for all clients, since one or more of them may have updated dues to pay for that month. This check happens everyday at midnight

INNOVATIVE FEATURE 4: ANALYSIS

GRAPHICAL ANALYSIS OF THE COMPLETE DATA BY SHAREHOLDRES



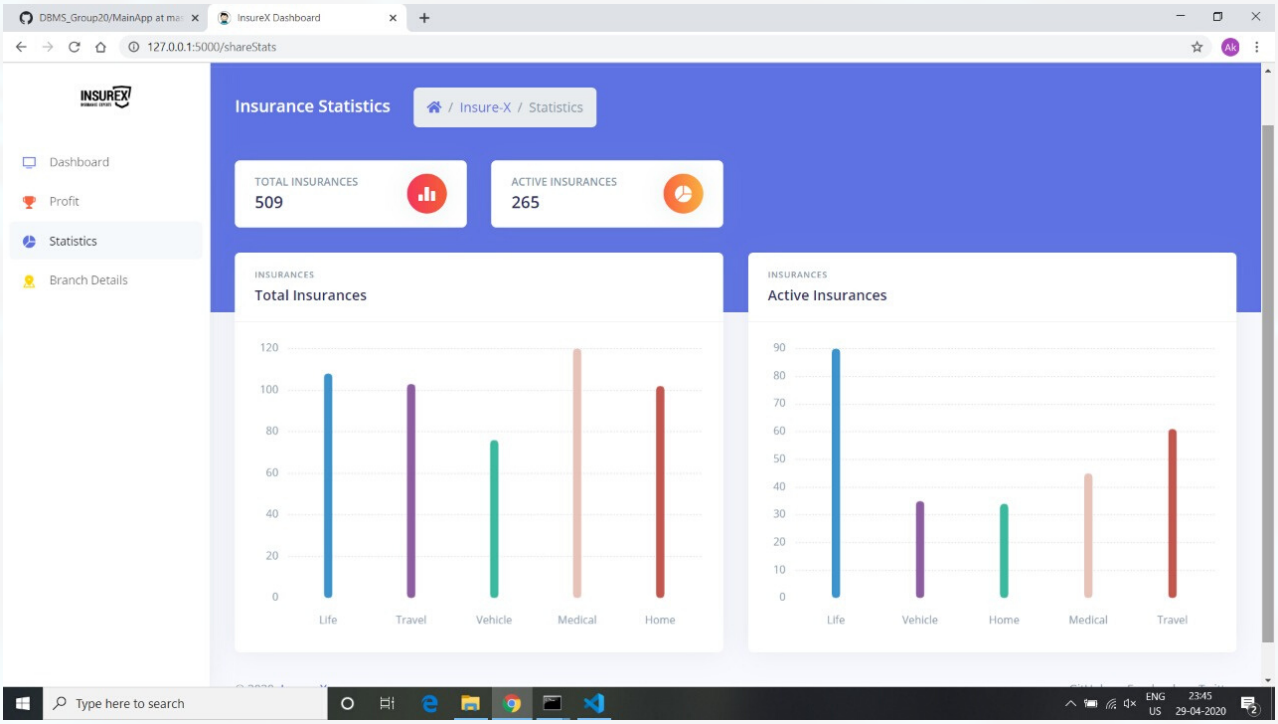
the shareholdres can view and analyse the profit and other aspects of the firm independently.

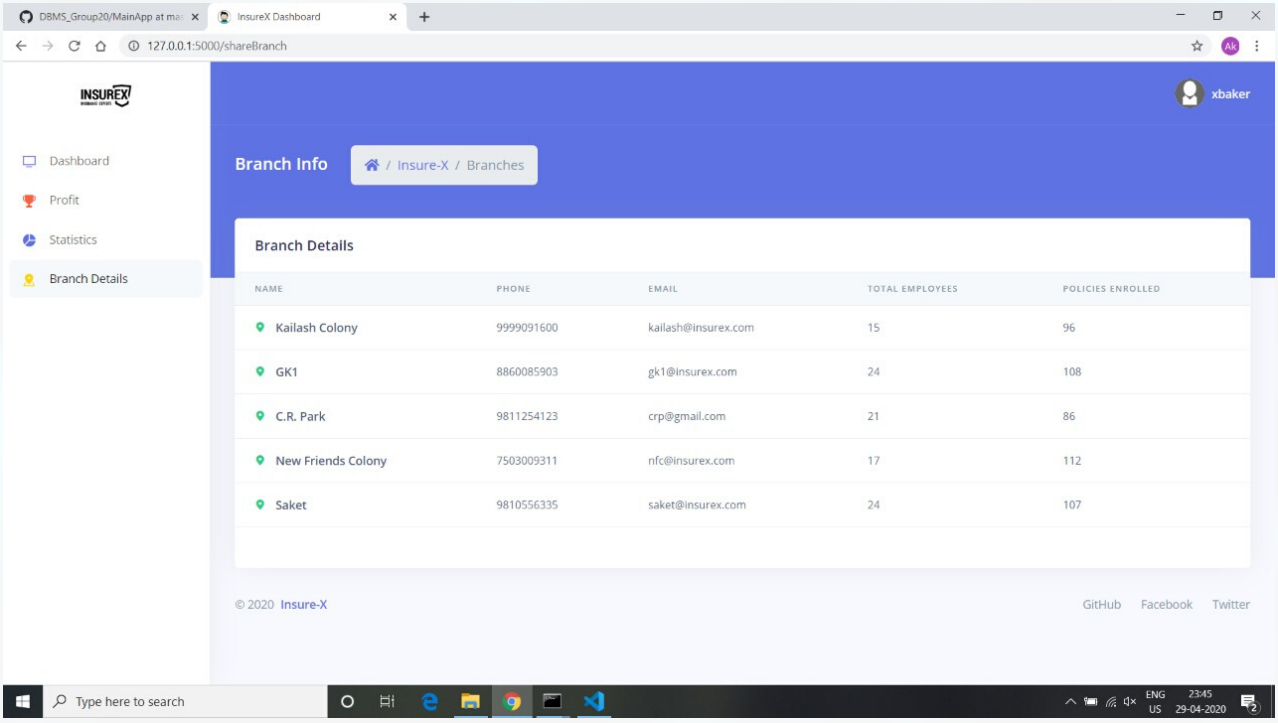
This is done as follows - The shareholdres logs into his or her own account, and then clicks to check the proft from the side bar. All these statistics are displayed in the form of bar graphs, which make analysing the data a lot easier.

More screenshots have been attached below.

INNOVATIVE FEATURE 4: ANALYSIS

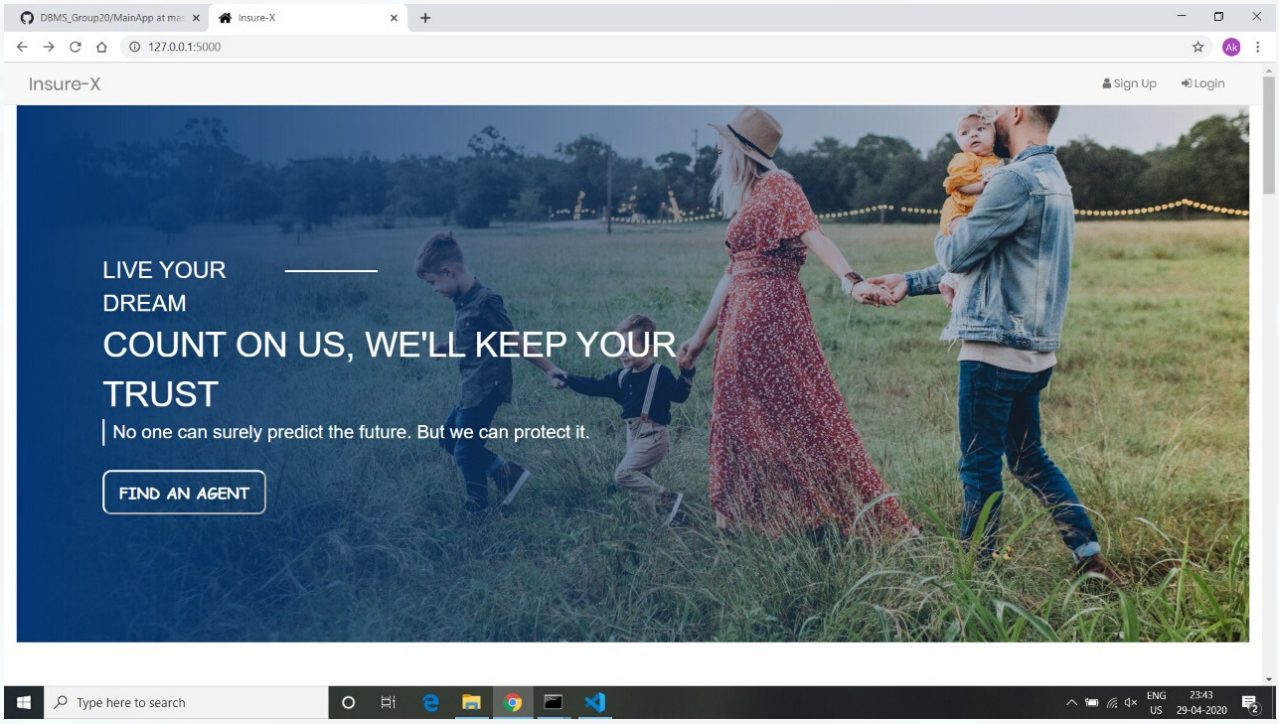
GRAPHICAL ANALYSIS OF THE COMPLETE DATA BY SHAREHOLDRES





INNOVATIVE FEATURE 5: USER-INTERFACE

A ROBUST USER-INTERFACE WITH WELL DESIGNED ELEMENTS AND INFORMATION ARCHITECTURE



The user interface of the web application supports clear and quick navigation, consistency and a brilliant user-experience.

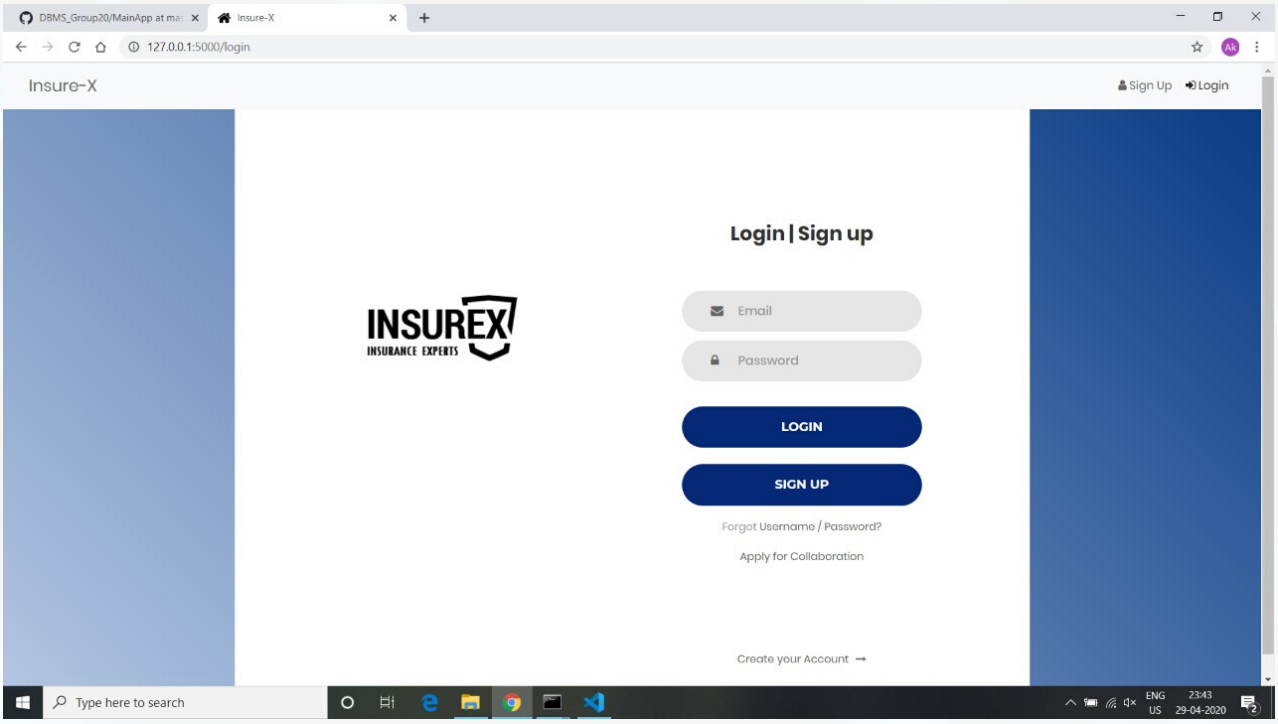
The interface reduces user-erros, has placeholder guides and a well-thought off informaiton architecture which is easy to understand.

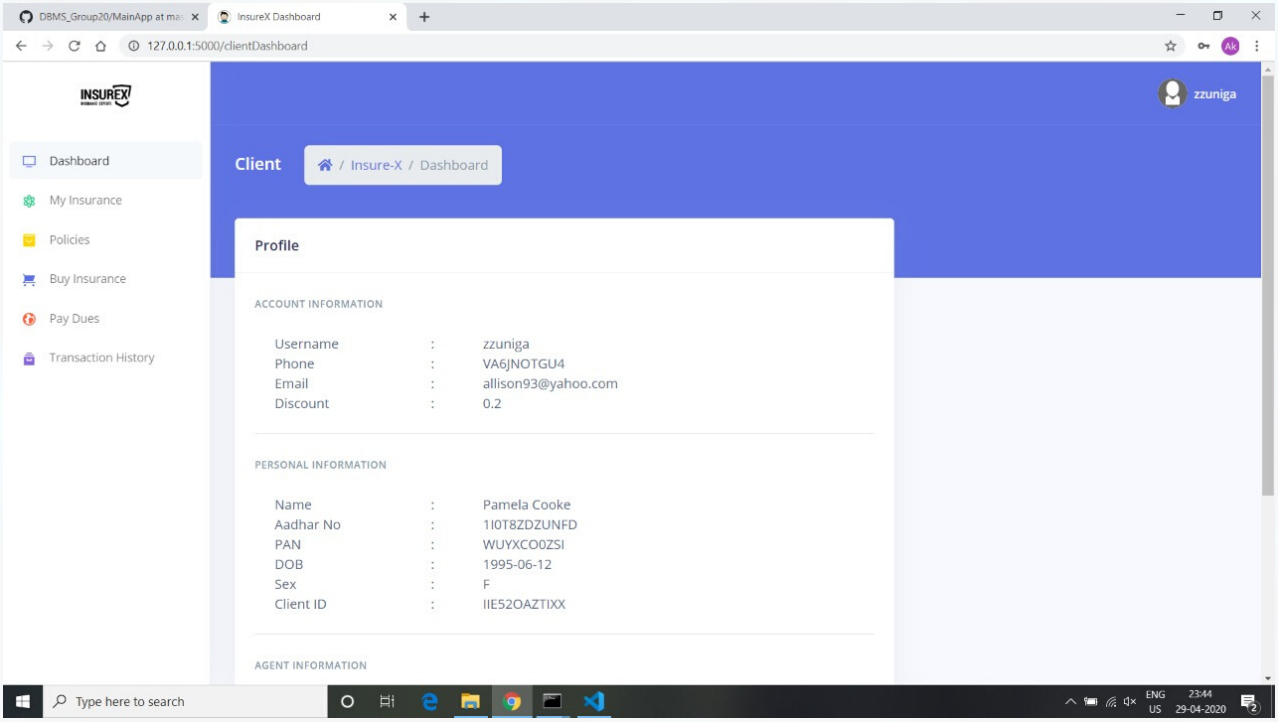
It not only seems visually appealing, but also fulfills the exact purpose, solving a major part of our problem statement.

The interface involves more than 10 screens, each made using HTML, CSS, Bootstrap and JavaScript. It is interactive yet minimalistic, and best for the user's needs.

INNOVATIVE FEATURE 5: USER-INTERFACE

A ROBUST USER-INTERFACE WITH WELL DESIGNED ELEMENTS AND INFORMATION ARCHITECTURE





FINAL WEB APPLICATION

CLICK ABOVE TO VIEW A FULL VIDEO DEMO

CONTRIBUTION

AYAAN KAKKAR

- Major Front-end screens using Bootstrap, HTML, CSS
- Used Flask to link Back-end and Front-end
- Embedded SQL queries
- Back-end

PRAGYAN MEHROTRA

- Database Design
- Data Population
- Embedded SQL Queries
- Back-end

KHUSHALI VERMA

- Work-in-Progress Document
- Database Design + ER Diagram
- Relational Queries

RITIK KHANNA

- Front-end implementation

AAKASH KUMAR SAGAR

- Helped in database designing