Assignment 1

CSL 332: Introduction to Database Systems

Md.H.Rahman- 2012cs50291

M.Samheeth- 2012cs10235

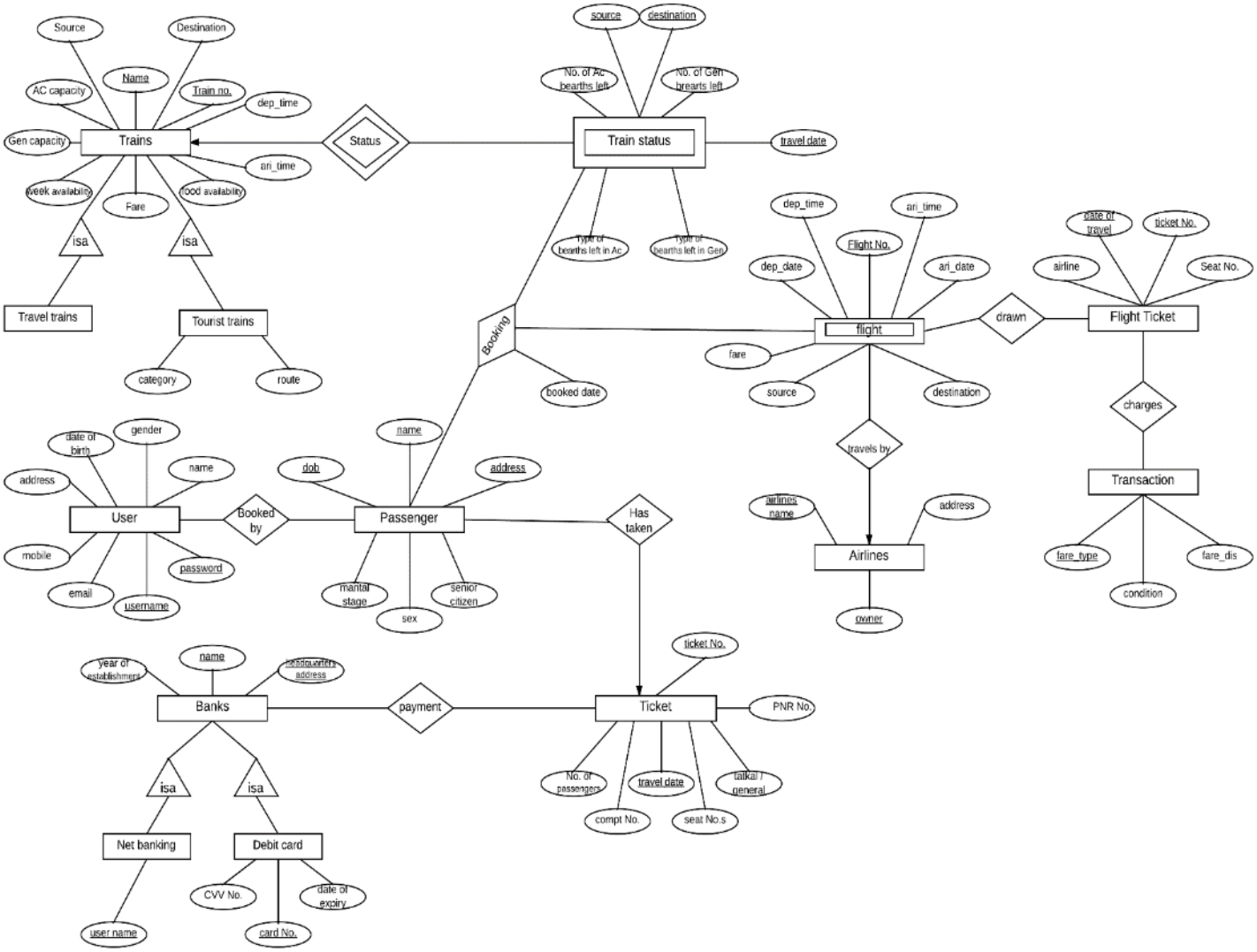
Contents

1. IRCTC Database Design

1.1 ER Model

Diagram

(2.5)



Description (1.5)

S.No	Entities	Attributes	Keys
1	Trains	Train_No., name, source, destination, AC capacity, Gen capacity, week availability, fare, food availability, dep_time, ari_time.	name, Train No.
2	Train status	Source, destination, Type of berths left in AC, type of berths left in Gen, No. of AC berths left, No. of Gen berths left, travel date.	booked date, travel date, Train_No., source, destination
3	User	name, gender, dob, address, mobile, email, username, password	Username, password
4	Passenger	name, address, dob, marital stage, sex, senior citizen	name, address, dob
5	Flight	Flight No., dep_date, dep_time, ari_date, ari_time, source, destination, fare.	Airlines name, flight No.
6	Airlines	Airlines name, owner, address	Airlines name, owner
7	Flight ticket	Ticket No., airline, date of travel, seat No.	Ticket No., date of travel
8	Transaction	fare_dis, fare type, condition	Fare type
9	Ticket	Ticket No., PNR No, travel date, seat No., cmpt No., quota, No. of passengers	Ticket No., travel date.
10	Banks	Name, head quarter address, year of establishment.	Name, head quarter address

1.2 Relational Model (3)

Trains (TrainNo. name, source, destination, AC capacity, Gen capacity, week availability, fare, food availability, dep_time, ari_time.)

Train status (Type of berths left in AC, type of berths left in Gen, No. of AC berths left, No. of Gen berths left, travel date, booked date, Train No., source, destination)

User (name, gender, DOB, address, mobile, email, username, password)

Passenger (name, address, DOB, marital stage, sex, senior citizen)

Flight (Flight No., dep_date, dep_time, ari_date, ari_time, source, destination, airlines name)

Flight ticket (Ticket No., airline, date of travel, seat no.)

Airlines (Airlines name, owner, address)

Transaction (fare_dis, fare type, condition)

Ticket (Ticket No., PNR No, travel date, seat No., cmpt No., quota, No. of passengers)

Banks (Name, head quarter address, year of establishment)

Booked by (Username, password, passenger name, address, DOB)

Booking (name, address, DOB, travel date, booked date)

Status (travel date, train name, train no.)

Has taken (name, address, DOB, travel date, ticket no.)

Payment (bank name, head quarter address, travel date, ticket no.)

Travels by (airlines name, owner, Flight no.)

taken (ticket no., date of travel, Flight no.)

charges (ticket no., fare type, date of travel)

1.3 Functional dependencies

(1)

1. Train status
   Booked date, travel date, Train_No , source, destination-> No. of AC berths left, No. of Gen berths left.
   Booked date, travel date, Train_No, source, destination, No. of AC berths left -> type of AC berths left.
   Booked date, travel date, Train_No, source, destination, No. of Gen berths left -> type of Gen berths left.
2. User
   Username, password -> email, mobile.
   Username, password, mobile -> name, address
   Username, password, name, address -> DOB, gender
3. Passenger
   DOB -> senior citizen.
   Name, DOB, address ->marital stage, sex.
4. Train
   Name, train no. -> Source, destination, AC capacity, Gen capacity, week availability, fare, food availability, dep_time, ari_time.
5. Flight
   Flight no., airlines name -> dep_date, dep_time, ari_date, ari_time, source, destination.
6. Flight ticket
   Ticket no., -> airline
   Ticket no., airline, date of travel ->seat no.
7. Airlines
   Airline name, owner ->address.
8. Transaction
   Fare type -> condition.
   Fare type, condition -> fare_dis.
9. Ticket
   Ticket no., date of travel -> PNR no., quota.
   Ticket no., PNR no., date of travel -> no. of passengers, seat no., comp no.

10. Banks

Name, head quarter address -> year of establishment

1.4 Other relations

(1.5)

Weak entity sets

S.no.	Weak entity sets
1	Flight
2	Train status

Non binary relationship

S.no.	relationships
1	booking

Hierarchical relationships

S.no	Parent entity	Child entities
1	trains	Tourist trains, travel trains
2	banks	Net banking, debit card

Constraints

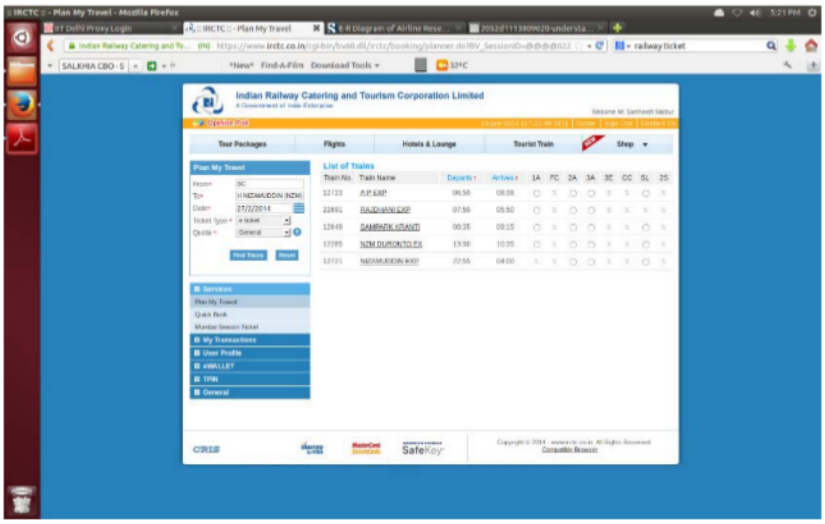
S.no.	Type of relation	From	To
1	Referential	Train status	trains
2	Referential	Flight	airlines
3	Many to one	Passengers	Tickets

1.5 Sample tuples

Trains

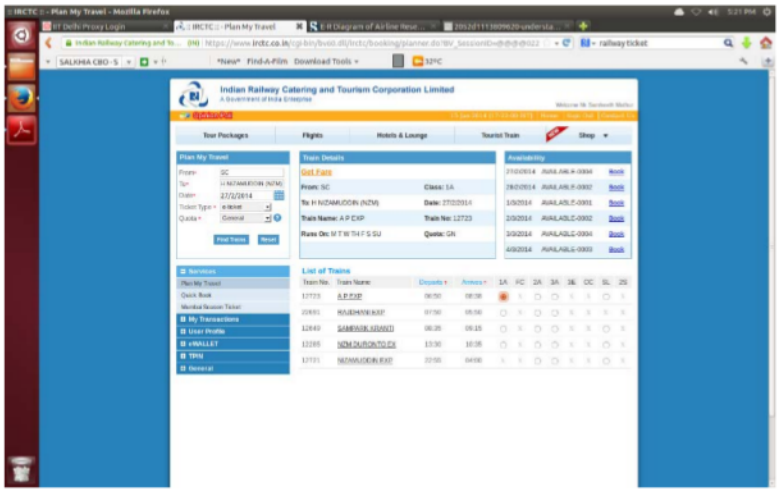
(2)

Name	Train no.	Source	Dest	Dep_ time	Ari_time	Week availability	Food availability	Ac	Gen	fare
AP exp	12723	SC	NZM	6:50	8:40	M,T,W,Th,F, Sa,S	Yes	300	700	1500
Rajdhani	22691	SC	NZM	7:50	5:50	W	Yes	250	600	2200



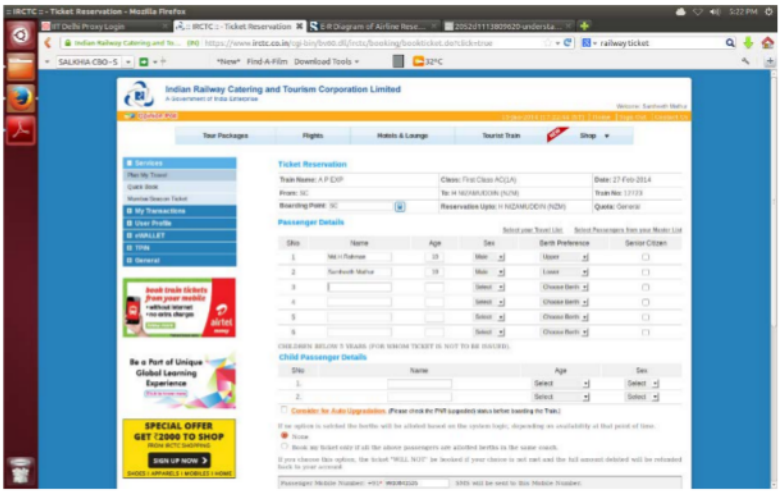
Train status

Train no	Travel date	Booked date	No of AC seats left	No of GEN seats left	Type of AC berths left	Type of Gen berths left	source	Dest
12723	27 feb 2014	15 jan 2014	4	46	L,U,SL	all	SC	NZM



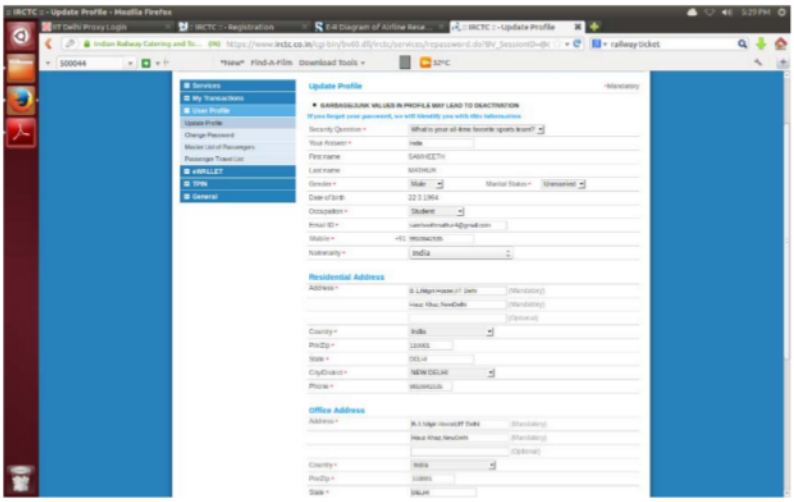
Passengers

Name	Address	Age	Marital stage	Sex	Senior citizen
Rahman	IIT-Delhi	19	No	Male	No
Samheeth	IIT-Delhi	19	No	Male	no



User

Name	DOB	Gender	Email	mobile	Address	Username	password
Samheeth	22-3-1994	Male	Sa....com	9910841535	b1,....,delhi	Samheeth3	Samheeth.3

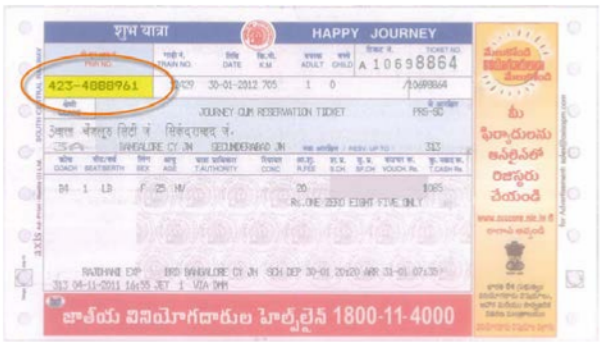


Ticket

Ticket No.	PNR No.	Quota	Seat No.	Travel date	Cmpt no.	No. of passengers
A10698864	423-4888961	Gen	1	30 jan 2012	B4	1

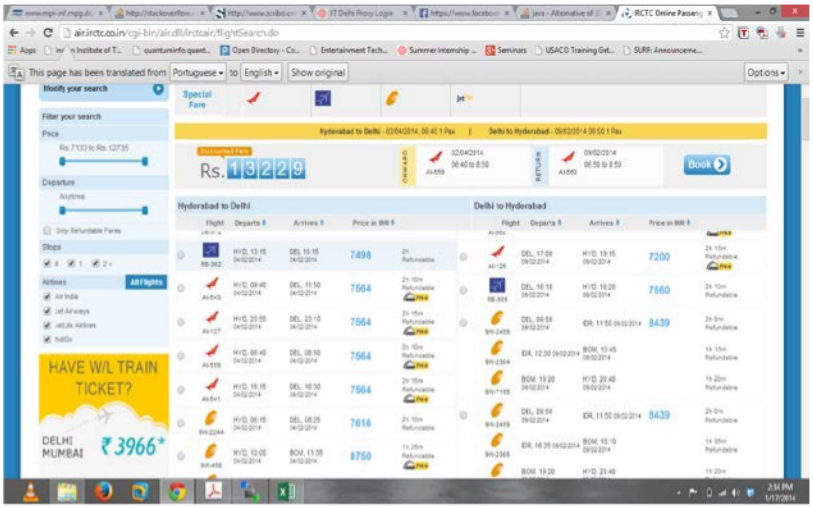
Flight Ticket

Ticket no.	Seat No.	Date of journey	Airline
Jl417	27k	26 july	Japan



Flights

Flight no.	Dep_time	Ari_time	Dep_date	Ari_date	Fare	Source	Destination
6E-302	13:15	15:15	4 feb 2014	4 feb 2014	7498	HYD	Delhi
AI-126	17:00	19:15	9 feb 2014	9 feb 2014	7200	Delhi	HYD



2. Decomposition

2.1. Universal relation
2.2. 2NF
2.3. 3NF
2.4. BCNF

2.1 Universal relation (4)

Username	Password	Email	Passenger name	Address	Passenger DOB	sex	Senior citizen	Ticket No.	Date of travel	Seat No.
samheeth3	samheeth.3	mathur@gmail.com	samheeth	b-1, Delhi	22 mar 1995	M	No	126	2 mar	34
ram	ram.5	ram@gmail.com	Latha	1-3-36/f , AP	21 jan 1995	F	No	132	3 nov	65
ravi	ravi.7	Ravi5@gmail.com	Latha	1-3-36/f ,MP	21 jan 1995	F	No	131	4 dec	34
ravi	ravi.7	Ravi5@gmail.com	Suma	D-26, jaya, UP	20 oct 1959	F	Yes	131	4 dec	34
Shane	Shane.3	shane@gmail.com	Rahul	TRR clg, AP	21 apr 1966	M	Yes	123	1 apr	45
Kranti	Kranti.3	raj@gmail.com	Raj	Vizag, AP	29 jun 1989	M	No	341	5 jan	89

2.2 2NF

Primary key for the above universal relation is the set (username, password, passenger name, address, DOB, ticket date, date of travel).

The FD's

1. username, password, address, DOB -> email
2. Passenger name, address, DOB, ticket no. -> sex, senior citizen
3. Ticket no., date of travel, username -> seat no.

These violates the 2NF because non-prime attributes can be derived from proper subsets of the candidate key.

These violation leads to the division of the universal relation table.

Username	Password	Email	AID
samheeth3	samheeth.3	mathur@gmail.com	1
ram	ram.5	ram@gmail.com	2
ravi	ravi.7	Ravi5@gmail.com	3
Shane	Shane.3	shane@gmail.com	4
Kranti	Kranti.3	raj@gmail.com	5

AID	ID
1	1
2	2
3	2
3	3
4	4
5	5

ID	Passenger name	Address	Passenger DOB	sex	Senior citizen	PID
1	samheeth	b-1, Delhi	22 mar 1995	M	No	1
2	Latha	1-3-36/f , AP	21 jan 1995	F	No	2
3	Suma	D-26, jaya, UP	20 oct 1959	F	Yes	3
4	Rahul	TRR clg, AP	21 apr 1966	M	Yes	4
5	Raj	Vizag, AP	29 jun 1989	M	No	5

PID	TID
1	1
2	2
2	3
3	3
4	4
5	5

TID	Ticket No.	Date of travel	Seat No.
1	126	2 mar	34
2	132	3 nov	65
3	131	4 dec	34
4	123	1 apr	45
5	341	5 jan	89

Redundancies that 2NF eliminates

Example 1 **:** The information in the tuples (ravi,ravi.7,ravi5@gmail.com,Latha ,……) and (ravi,ravi.7,ravi5@gmail.com,Suma,……) booking a ticket appears 2 times in our universal relation but the division of the universal relation into sub-relations eliminates the repetition.

Example 2 : The information in the tuples (ram,…,Latha,1-3-36/f AP,21 jan 1995,….) and (ravi,…., Latha,1-3-36/f AP,21 jan 1995,……) booking a ticket appears 2 times in our universal relation but the division of the universal relation into sub-relations eliminates the repetition.

2.3 3NF

3NF violation implies

1. 2NF violation.
2. If X->Y is a non-trivial FD then either X is a super key or Y is prime.

After the division of universal table into sub-tables check for the 3NF violation.

In second table consider the FD (DOB-> senior citizen)

1. Senior citizen is not a prime attribute.
2. DOB is not the super key of that table.

Hence it violates the 3NF.

The universal relation after taking into consideration of 3NF and 2NF.

Username	Password	Email	AID
samheeth3	samheeth.3	mathur@gmail.com	1
ram	ram.5	ram@gmail.com	2
ravi	ravi.7	Ravi5@gmail.com	3
Shane	Shane.3	shane@gmail.com	4
Kranti	Kranti.3	raj@gmail.com	5

AID	ID
1	1
2	2
3	2
3	3
4	4
5	5

ID	Passenger name	Address	Passenger DOB	DID
1	samheeth	b-1, Delhi	22 mar 1995	1
2	Latha	1-3-36/f , AP	21 jan 1995	2
3	Suma	D-26, jaya, UP	20 oct 1959	3
4	Rahul	TRR clg, AP	21 apr 1966	4
5	Raj	Vizag, AP	29 jun 1989	5

DID	SID
1	1
2	2
3	3
4	4
5	5

sex	Senior citizen	PID
M	No	1
F	No	2
F	Yes	3
M	Yes	4
M	No	5

PID	TID
1	1
2	2
2	3
3	3
4	4
5	5

TID	Ticket No.	Date of travel	Seat No.
1	126	2 mar	34
2	132	3 nov	65
3	131	4 dec	34
4	123	1 apr	45
5	341	5 jan	89

2.4 BCNF

As the universal relation violates 3NF it also violates BCNF because BCNF only requires X to be a superkey in a FD(X->Y).

3. Datasets of smaller sizes

3.1 Required Schema

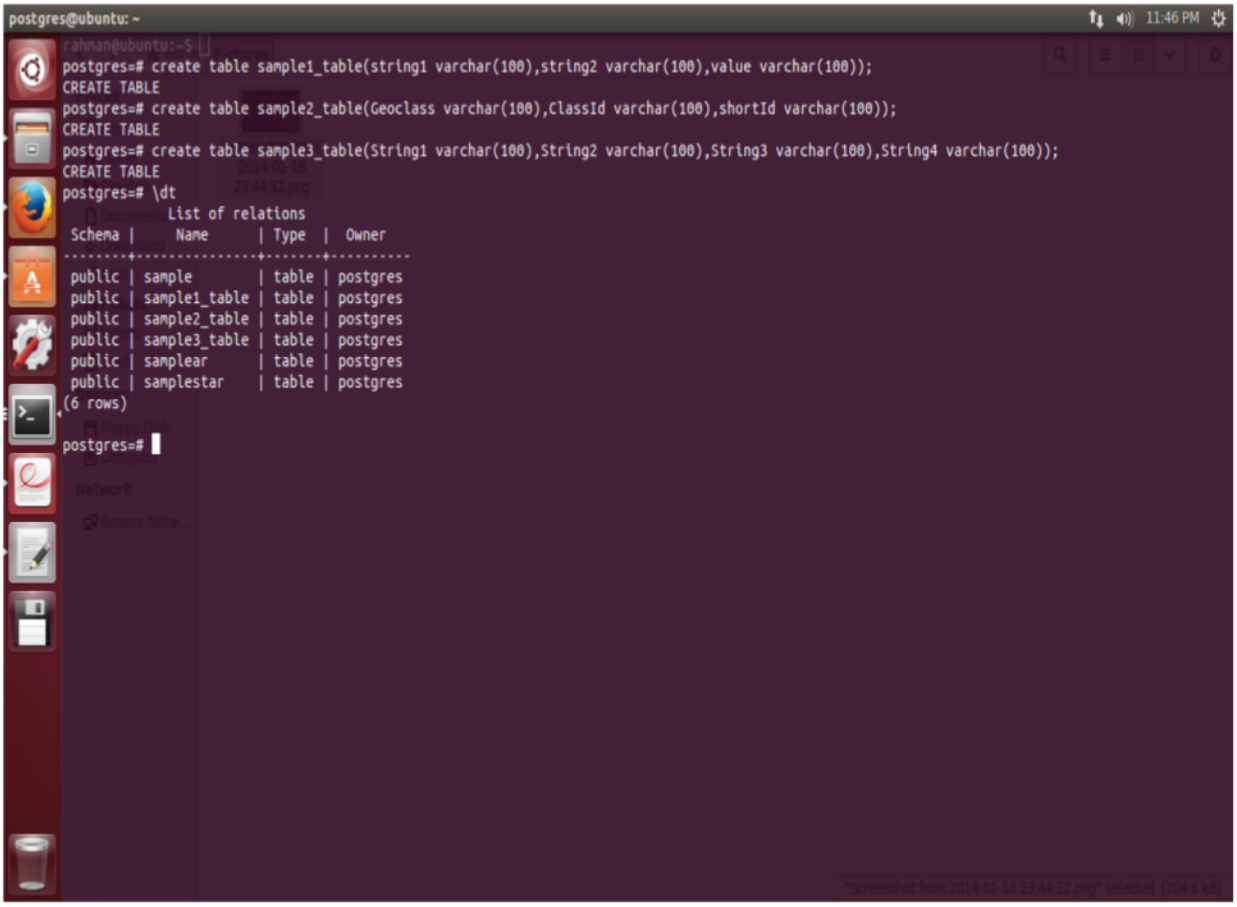
The datasets downloaded directly were replicated in csv format so that the total number of tuples in the csv file is between 50000 to 100000.The files consisted of 56678 , 57843 and 86663 tuples before they were inserted into the database.

For the three datasets obtained,three separate tables were created:

CREATE TABLE sample1_table(string1 varchar(100),string2 varchar(100),value varchar(100));

CREATE TABLE sample2_table(Geoclass varchar(100),ClassId varchar(100),shortId varchar(100));

CREATE TABLE sample3_table(String1 varchar(100),String2 varchar(100),String3 varchar(100),String4 varchar(100));



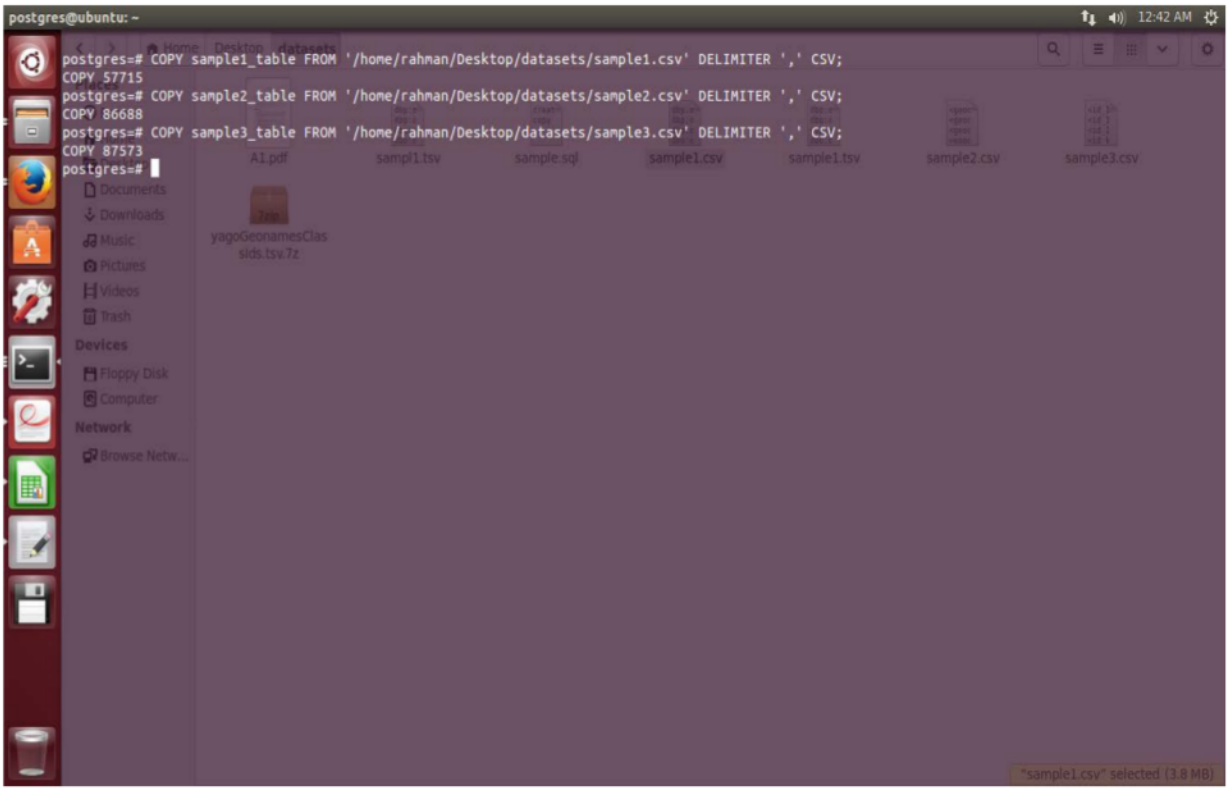
3.2.1 Bulk Load Insertion

The initial tables created are then populated with the datasets using:

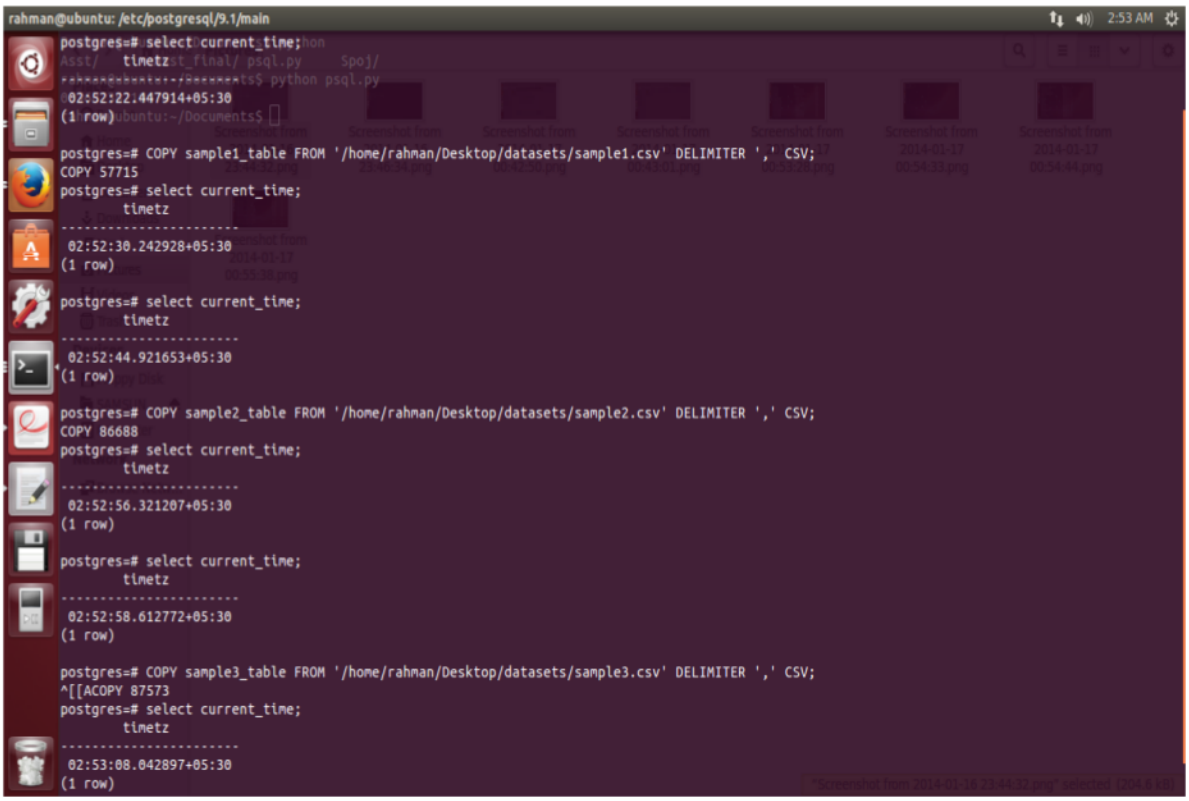
COPY sample1_table FROM ‘/home/rahman/Desktop/datasets/sample1.csv’ DELIMITER ‘,’ CSV;

COPY sample2_table FROM ‘/home/rahman/Desktop/datasets/sample2.csv’ DELIMITER ‘,’ CSV;

COPY sample3_table FROM ‘/home/rahman/Desktop/datasets/sample3.csv’ DELIMITER ‘,’ CSV;



The time taken for bulk load is computed from time stamp generated using SELECT current_time;



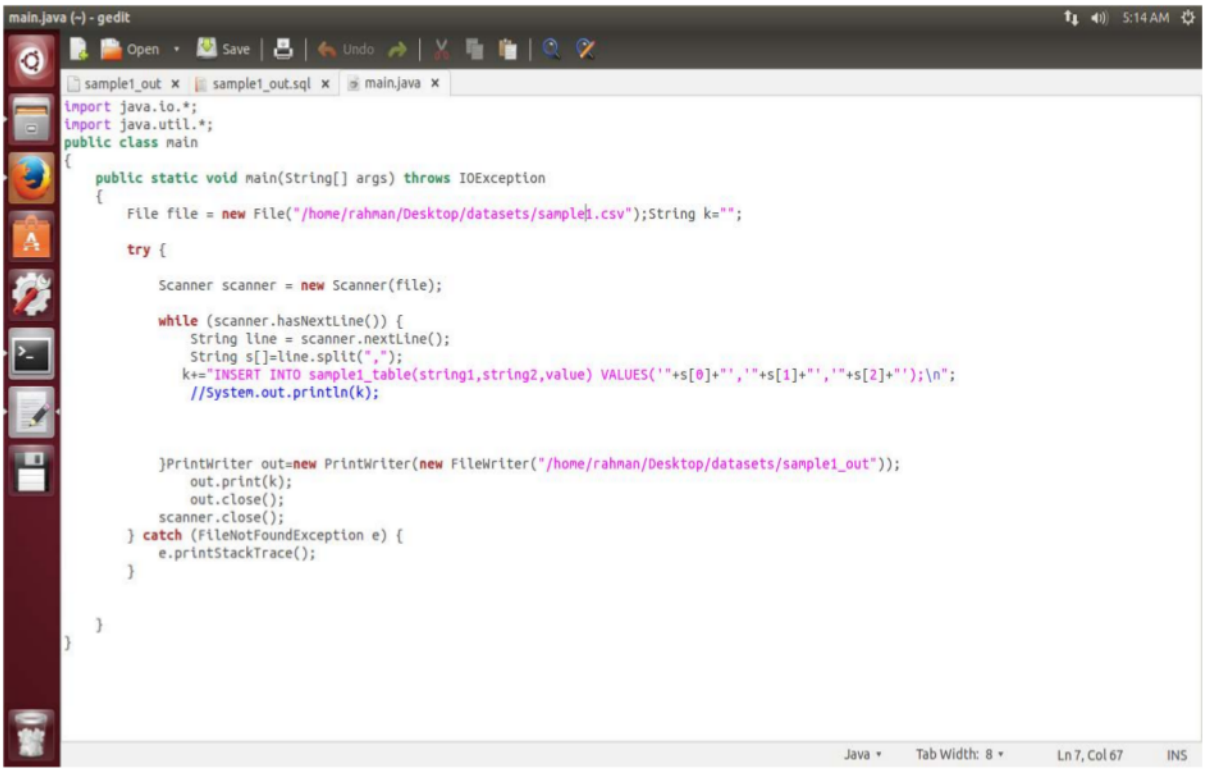
3.2.2 Insert each tuple

A piece of Java code is utilized to parse each line in csv file and generate the queries which are written to a sql file called sample1_out.sql and then later run it in psql.

The java code used is provided :

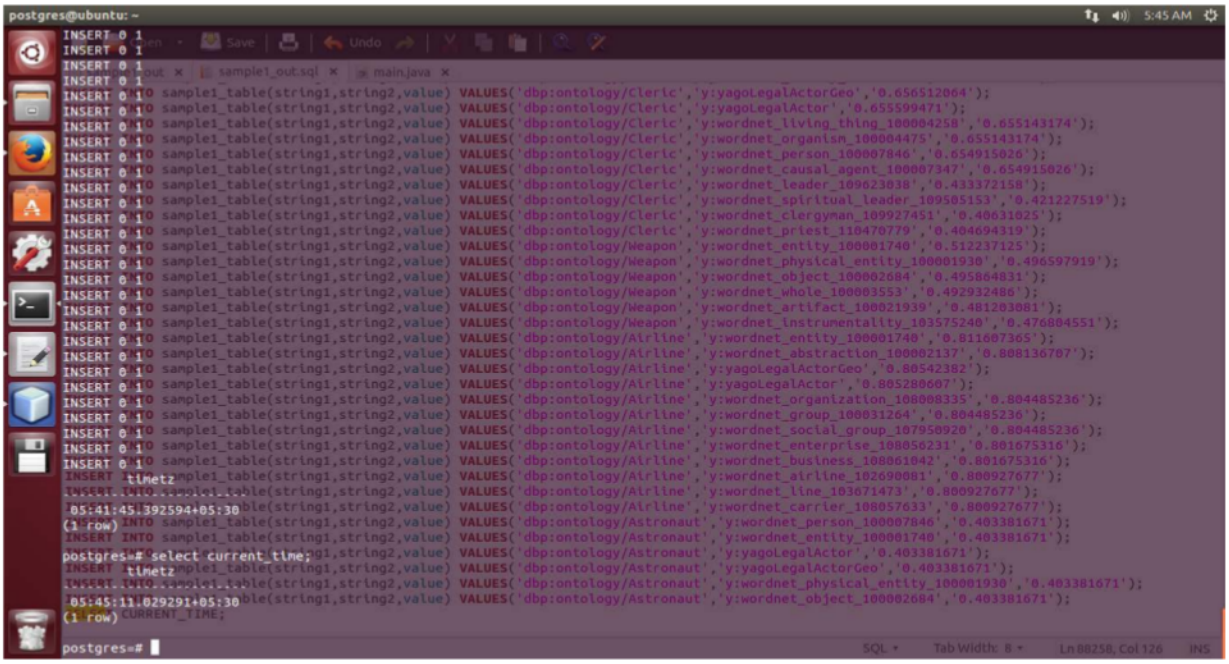
(2)

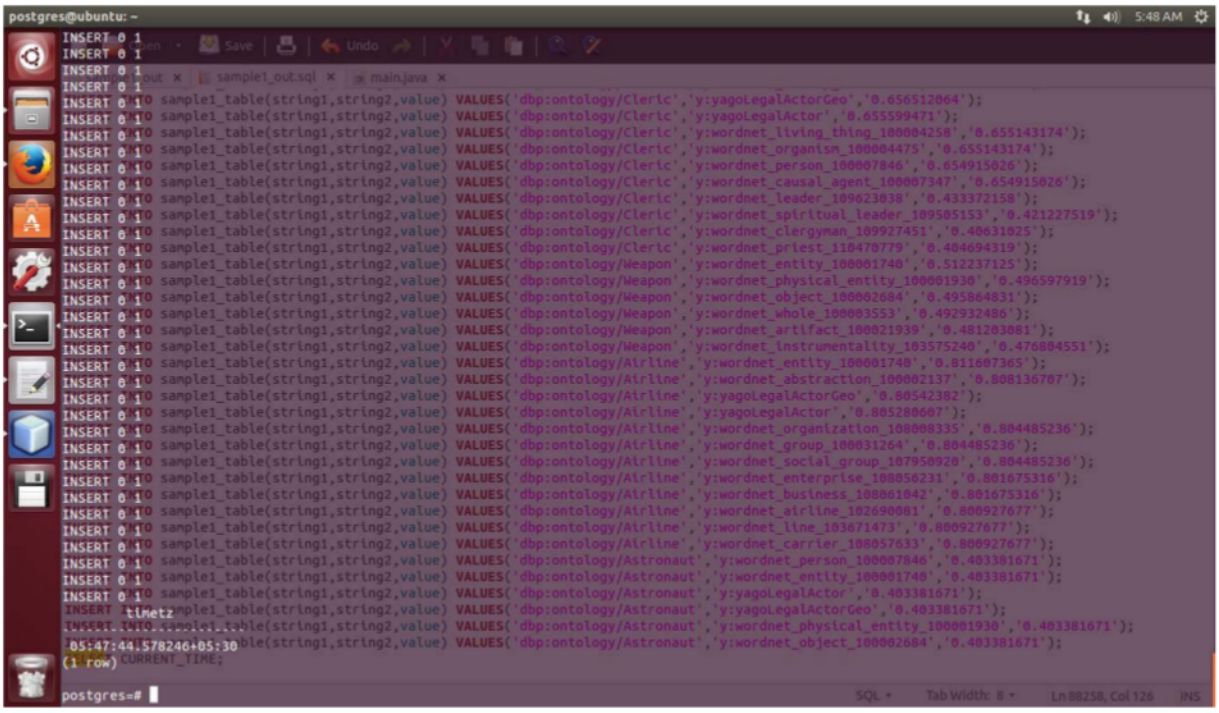
(jdbc had to be used)



Run the file 'sample1_out.sql' in psql using " \i sample1_out.sql" to insert each tuple in to the table in each of the 3 tables.

The time taken is computed from time stamps written at the beginning and end of the sql file.

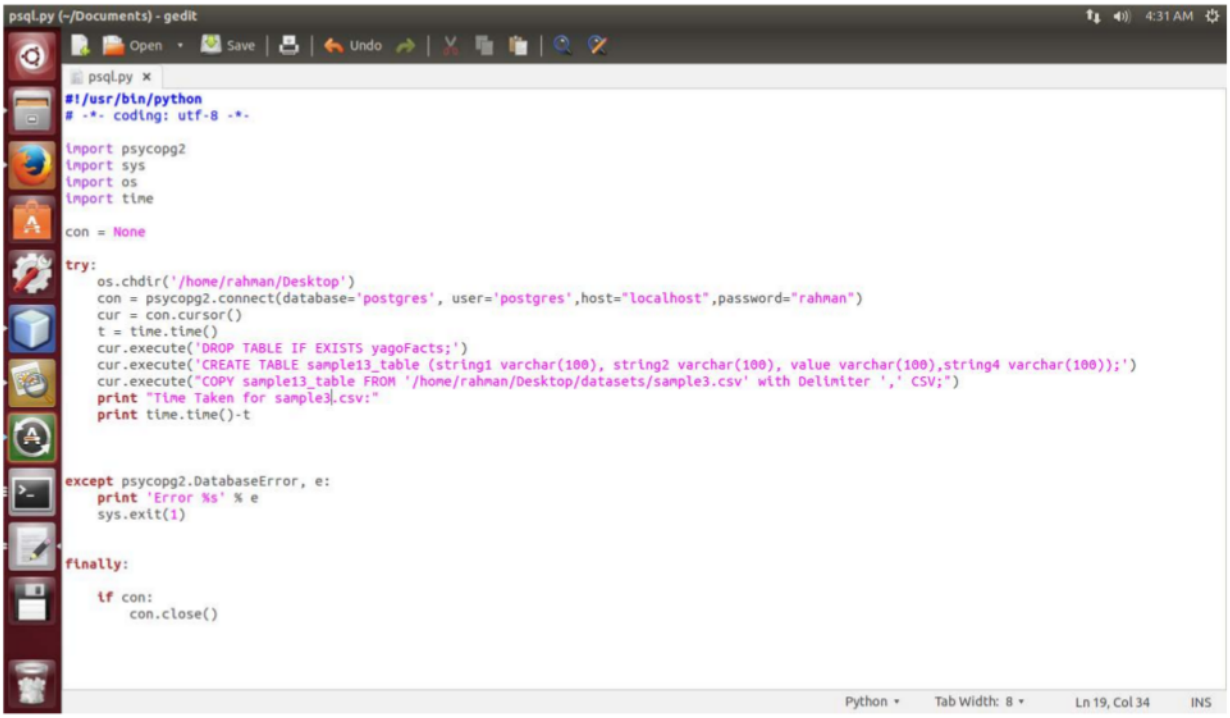




3.2.3 Using programmatic method

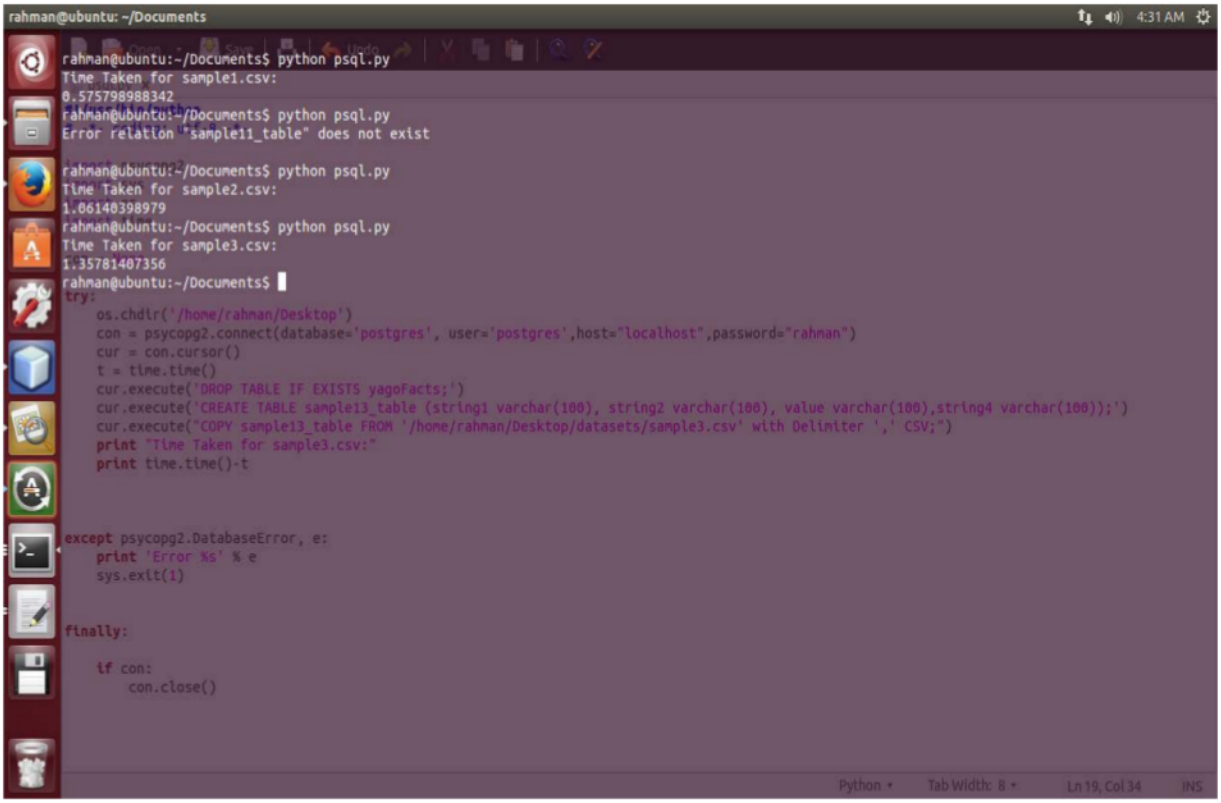
JDBC is not well compatible on Ubuntu machine hence psycopg2 (built on python) was used to communicate with the database.

The python script used is displayed below:



The time for these insert operations was computed from the python script using the inbuilt python function 'time.time()'.The difference in the times at the start and the end of the operation is printed on console.

The output thus generated when tables are populated for the 3 different datasets is provided below:



3.3 Other Required Details

Name of dataset	Size of dataset	Time for bulk load	Time for insert operations	Time taken when psycopg2 used
sample1.csv	3.8MB	7.802 sec	1 min 48 sec	0.5757 sec
sample2.csv	4.5MB	9.430 sec	2 min 33 sec	1.061 sec
sample3.csv	8.8MB	11.400 sec	2 min 52 sec	1.357 sec

(3.5)
(jdbc had to be used)

Configuration of the machine used :

- Device : Ubuntu 13.04
- Memory : 3.9 GB
- Processor : Intel® Core™ i7-3632QM CPU @ 2.20GHz
- Graphics : Gallium 0.4 on SVGA3D; build: RELEASE;
- OS type : 32-bit
- Disk : 80.2 GB

4. Datasets of larger sizes

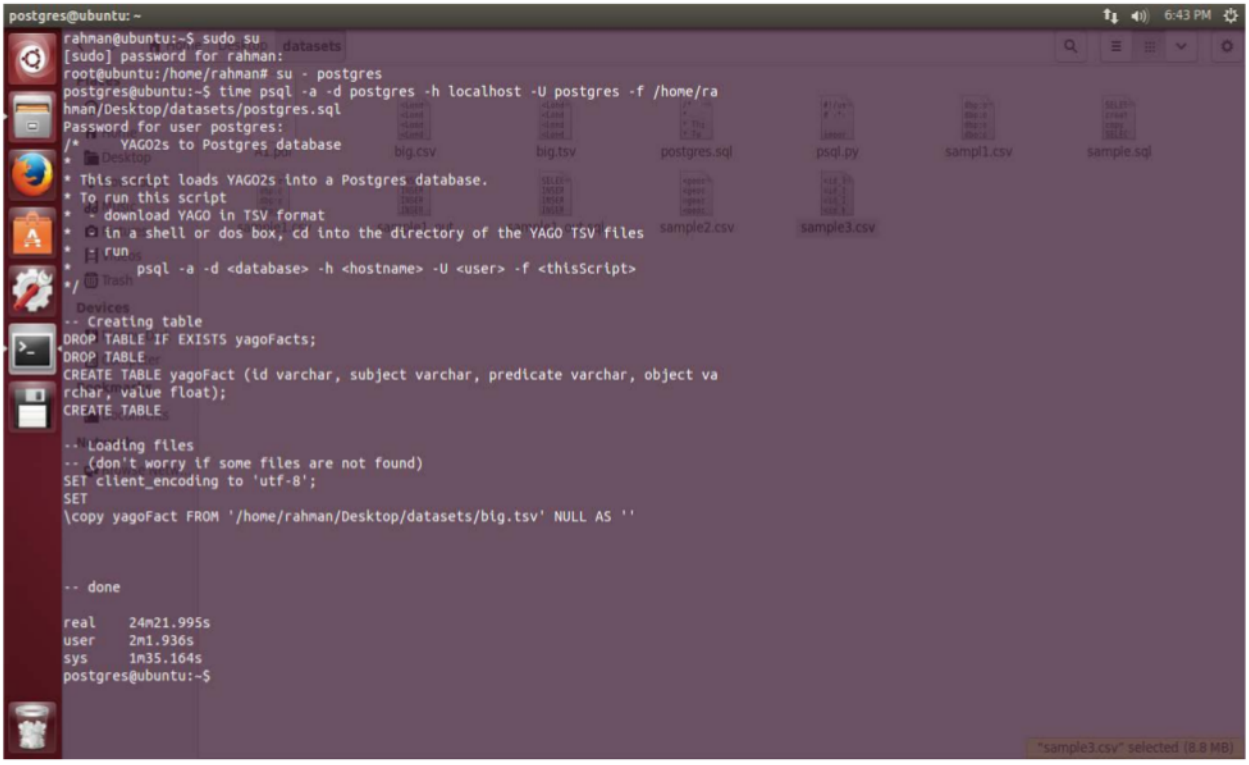
4.1 Bulk Load Insertion

A dataset of size 6.1 GB obtained by extracting a file of size 1.61 GB was used for the various operations.

The script used for bulk load insert is :

time psql –a –d postgres –h localhost –U postgres –f /home/rahman/Desktop/datasets/postgres.sql

The time taken for this operation is : 24 min 21.995 sec



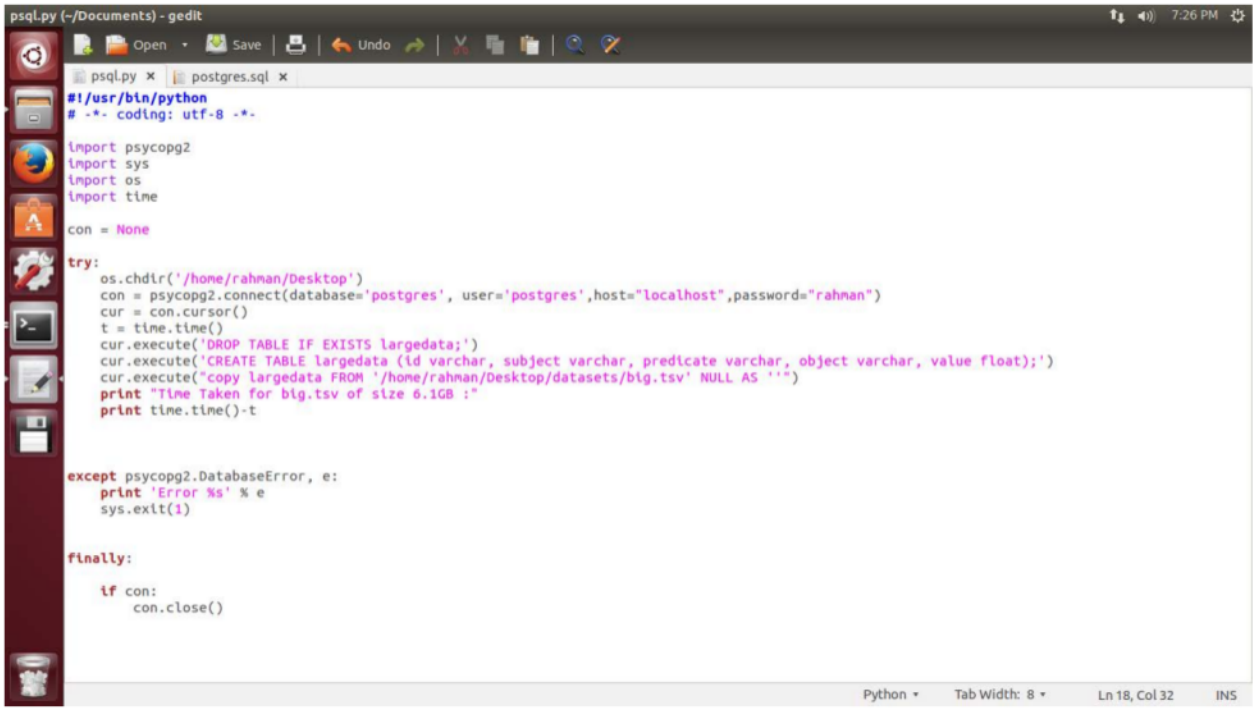
4.2 Insert each tuple

This operation takes a lot of time,both during the .sql file generation using the java program mentioned earlier.This takes around 75 minutes.

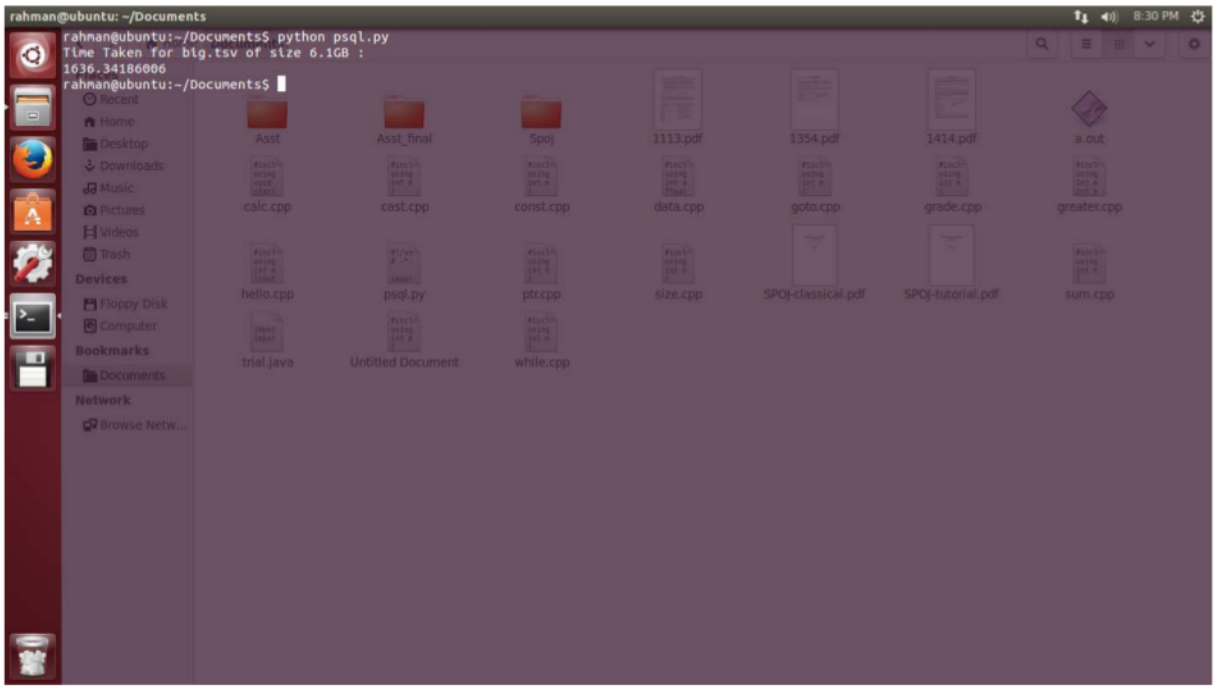
When the produced .sql file of size 9.7 GB is executed using the command '\i <path to .sql file>' ,several errors like 'out of memory' and others are seen and this method of populating the database is not preferable and theoretically takes a very large amount of time due to the parsing and inserting operations for each tuple in the table.

4.3 Using programmatic method

psycopg2 is used again for this method of populating the table. The script which is used for the dataset is shown below:



The time taken for this operation is taken note of , with the help of inbuilt python function ‘time.time()’ which returns the time taken in milliseconds.The time taken for this population is 27.27 minutes.



Files of size around 5GB to 10GB have more number of tuples than that could be accommodated by an Excel file.An Excel file can accommodate a maximum of 10,00,000 tuples.

Name of dataset	Size of dataset	Time for bulk load	Time for insert operations	Time taken when psycopg2 used
Big.tsv	6.1GB	24 min 21.995 sec	----------	27.27 min

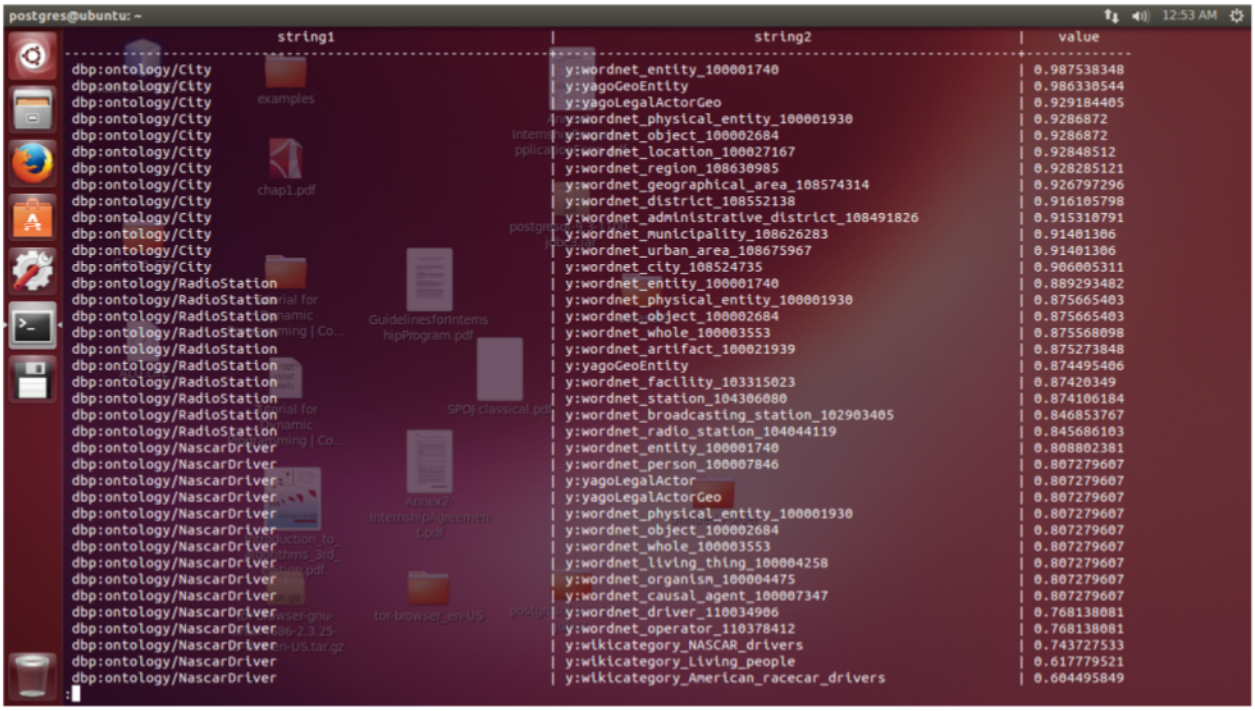
(2)

(incomplete answer and also jdbc required)

Any table is shown with all its constituent tuples by the command

‘select * from <table_name>;’

An example of such a view generated upon execution of this command is shown below:



string1	string2	value
dbp:ontology/City	y:wordnet_entity_100001740	0.987538348
dbp:ontology/City	y:yagoGeoEntity	0.986330544
dbp:ontology/City	y:yagoLegalActorGeo	0.929184405
dbp:ontology/City	y:wordnet_physical_entity_100001930	0.9286872
dbp:ontology/City	y:wordnet_object_100002684	0.9286872
dbp:ontology/City	y:wordnet_location_100027167	0.92848512
dbp:ontology/City	y:wordnet_region_108630985	0.928285121
dbp:ontology/City	y:wordnet_geographical_area_108574314	0.926797296
dbp:ontology/City	y:wordnet_district_108552138	0.916105798
dbp:ontology/City	y:wordnet_administrative_district_108491826	0.915310791
dbp:ontology/City	y:wordnet_municipality_108626283	0.91401306
dbp:ontology/City	y:wordnet_urban_area_108675967	0.91401306
dbp:ontology/City	y:wordnet_city_108524735	0.906005311
dbp:ontology/RadioStation	y:wordnet_entity_100001740	0.889293482
dbp:ontology/RadioStation	y:wordnet_physical_entity_100001930	0.875665403
dbp:ontology/RadioStation	y:wordnet_object_100002684	0.875665403
dbp:ontology/RadioStation	y:wordnet_whole_100003553	0.875568098
dbp:ontology/RadioStation	y:wordnet_artifact_100021939	0.875273848
dbp:ontology/RadioStation	y:yagoGeoEntity	0.874495406
dbp:ontology/RadioStation	y:wordnet_facility_103315023	0.87420349
dbp:ontology/RadioStation	y:wordnet_station_104306080	0.874106184
dbp:ontology/RadioStation	y:wordnet_broadcasting_station_102903405	0.846853767
dbp:ontology/RadioStation	y:wordnet_radio_station_104044119	0.845686103
dbp:ontology/NascarDriver	y:wordnet_entity_100001740	0.808802381
dbp:ontology/NascarDriver	y:wordnet_person_100007846	0.807279607
dbp:ontology/NascarDriver	y:yagoLegalActor	0.807279607
dbp:ontology/NascarDriver	y:yagoLegalActorGeo	0.807279607
dbp:ontology/NascarDriver	y:wordnet_physical_entity_100001930	0.807279607
dbp:ontology/NascarDriver	y:wordnet_object_100002684	0.807279607
dbp:ontology/NascarDriver	y:wordnet_whole_100003553	0.807279607
dbp:ontology/NascarDriver	y:wordnet_living_thing_100004258	0.807279607
dbp:ontology/NascarDriver	y:wordnet_organism_100004475	0.807279607
dbp:ontology/NascarDriver	y:wordnet_causal_agent_100007347	0.807279607
dbp:ontology/NascarDriver	y:wordnet_driver_110034906	0.768138081
dbp:ontology/NascarDriver	y:wordnet_operator_110378412	0.768138081
dbp:ontology/NascarDriver	y:wikicategory_NASCAR_drivers	0.743727533
dbp:ontology/NascarDriver	y:wikicategory_Living_people	0.617779521
dbp:ontology/NascarDriver	y:wikicategory_American_racecar_drivers	0.604495849