Presented By:

Krutik Parikh

# Task1:

## In order to create this task we have use talend software and created the address table in oracle database

**create table** Address**(**

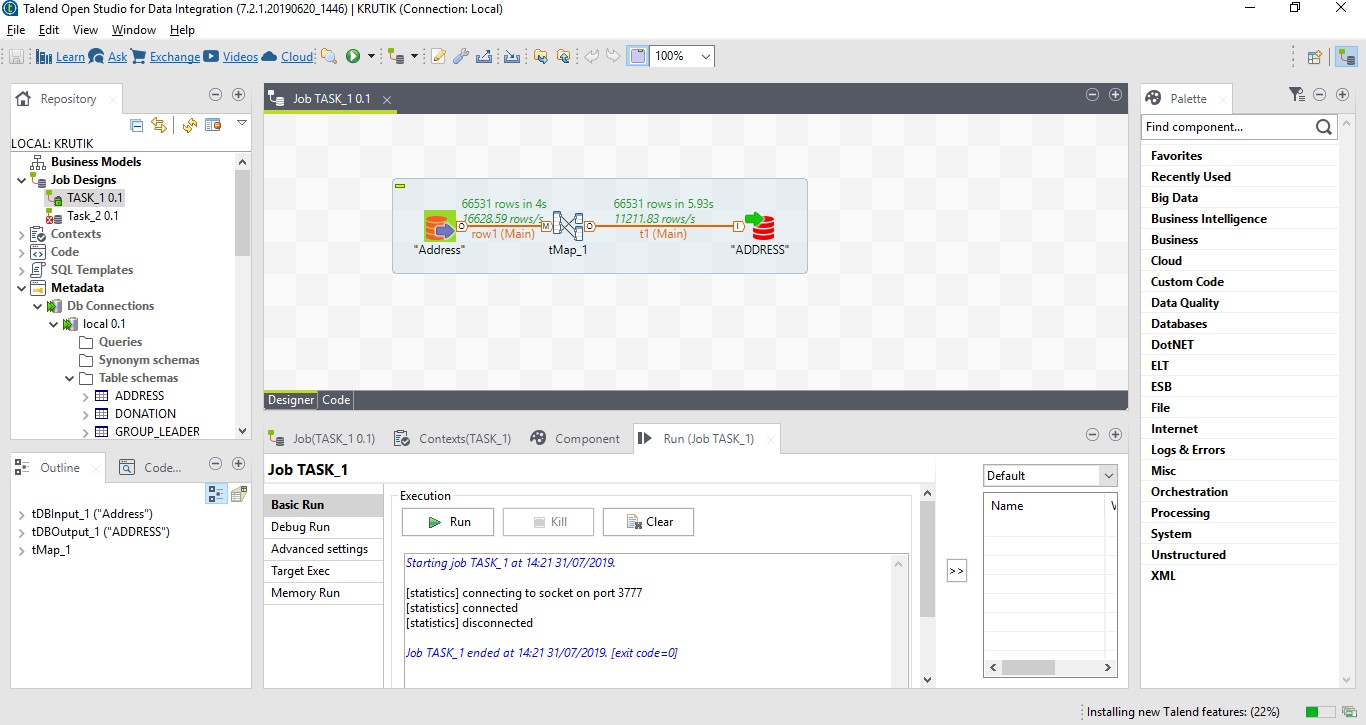
Address\_id **number primary key null,**

Unit\_num **varchar2(**6**),** Street\_number **number not null,** Street\_name **varchar2(**24**) not null,** Street\_type **varchar2(**12**) null,** Street\_description **char(**1**),** Postal\_code **char(**7**) null,**

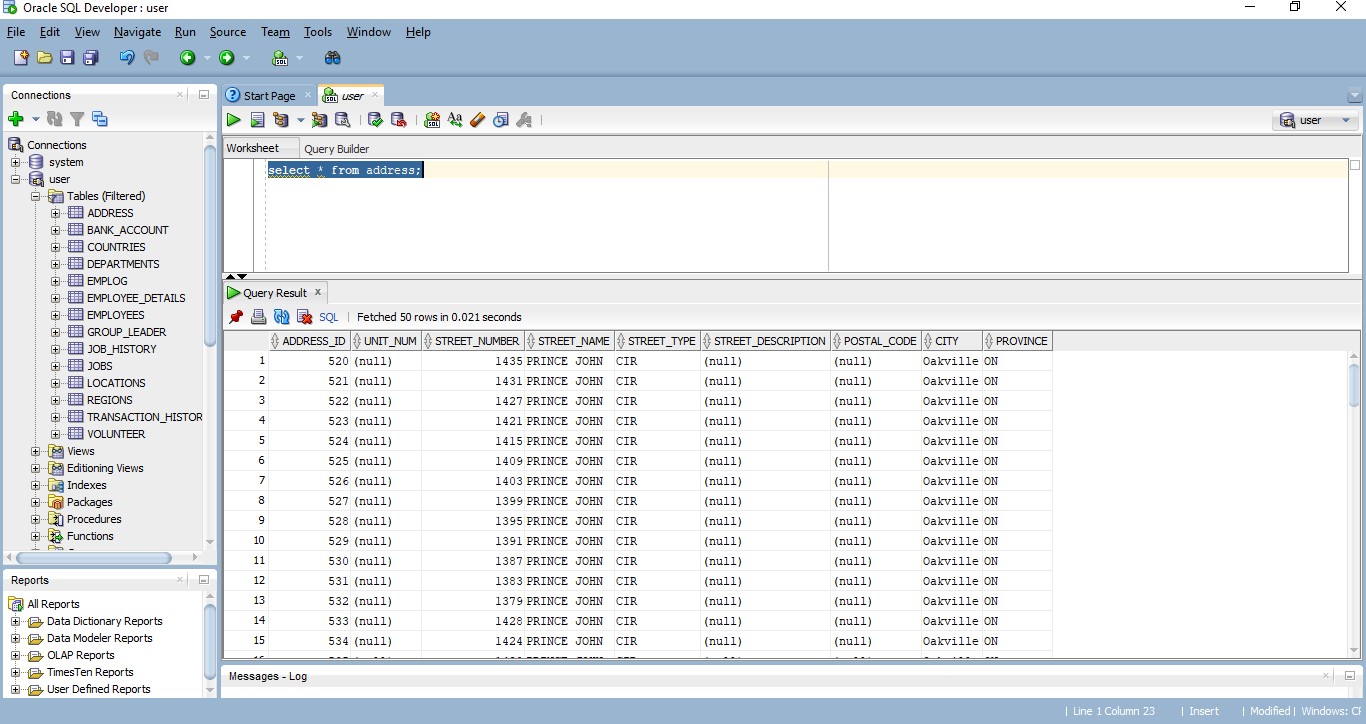
City **varchar2(**16**) not null,** Province **char(**2**) not null, constraint** st\_dir **check**

**(**Street\_description **in(**'E'**,**'W'**,**'N'**,**'S'**)));**

## In order to do this we have created the connecting tmap to Sheridan database and local address table for output.



After successfully loading data to our local database.



**Task2**:

We are getting so many errors in talend so we have tried loading data using PL/SQl traditional method.

[**https://www.youtube.com/watch?v=yD\_CL\_cf8tQ**](https://www.youtube.com/watch?v=yD_CL_cf8tQ)

#### The code that is used to load data from donations.txt to donation.stg(I have entered one more row so that after I can check for invalid address)

LOAD **DATA**

INFILE 'C:\Users\Krutik\Desktop\DBMS\DBMS\Donations.txt'

##### TRUNCATE

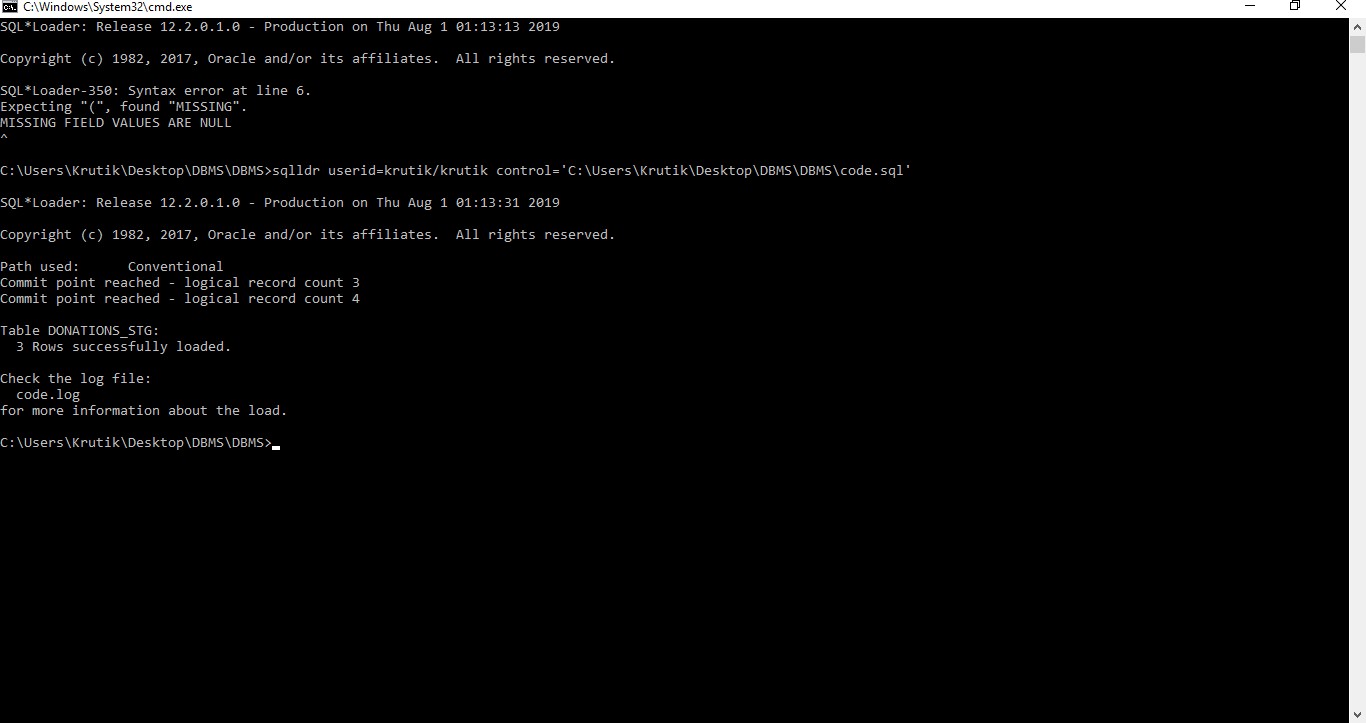
**INTO TABLE** DONATIONS\_STG FIELDS TERMINATED **BY** ',' **(**

Address**,** DonorName**,** DonationDate**,** DonationTime**,** Amount**,** DonationType**,** VolunteerId

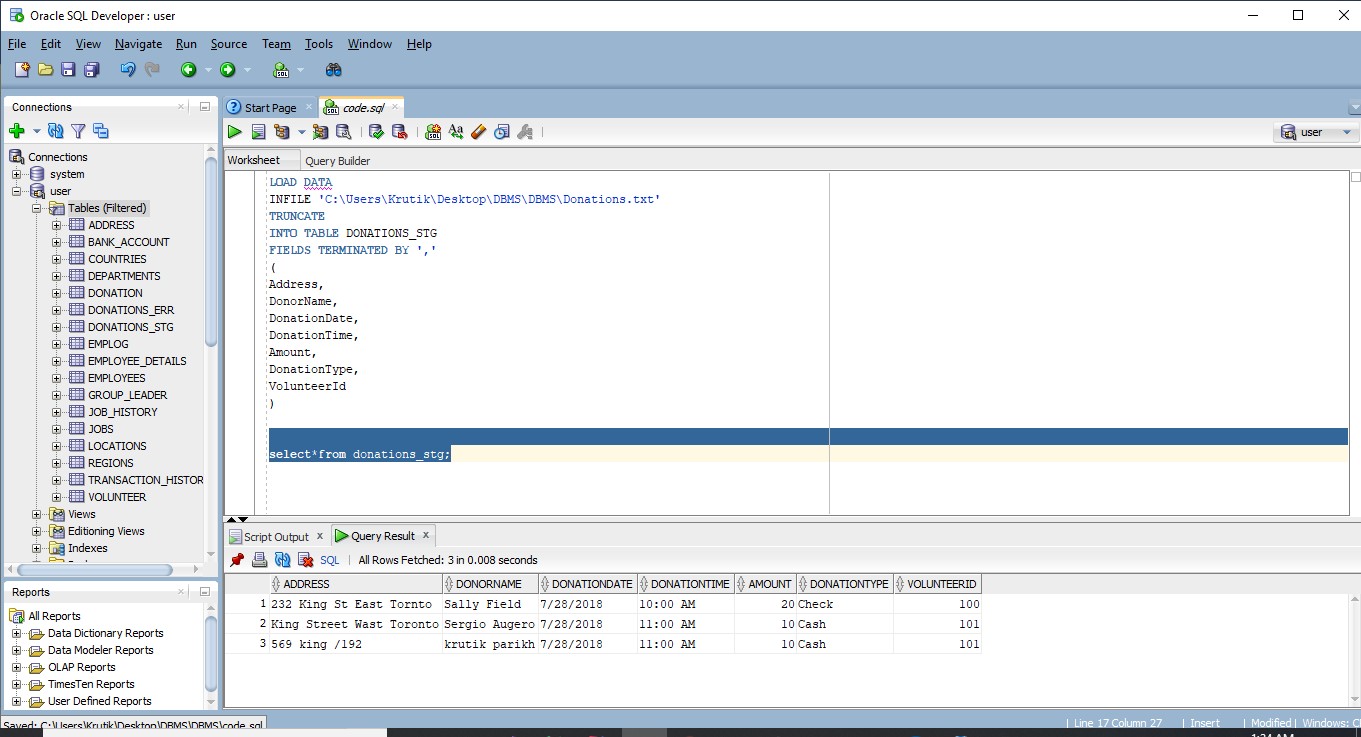
**)**

So to run this we need to save the query and execute from cmd (choose the path where the donation file is saved it is also used to load csv file) and just write

Localpath**//**sqlldr userid**=**username**/**password control**=**’**(**The **path where** you have save the **sql** query **to execute)**’



## Here is the select statement for donations\_stg table



**Task3:**

So before starting and creating the procedure we need to make the default directory that help the data to store in preferred location like this

**-- Create directory**

**create or replace** directory MF\_KRUTIK\_DIR1

**as** 'D:\Krutik'**;**

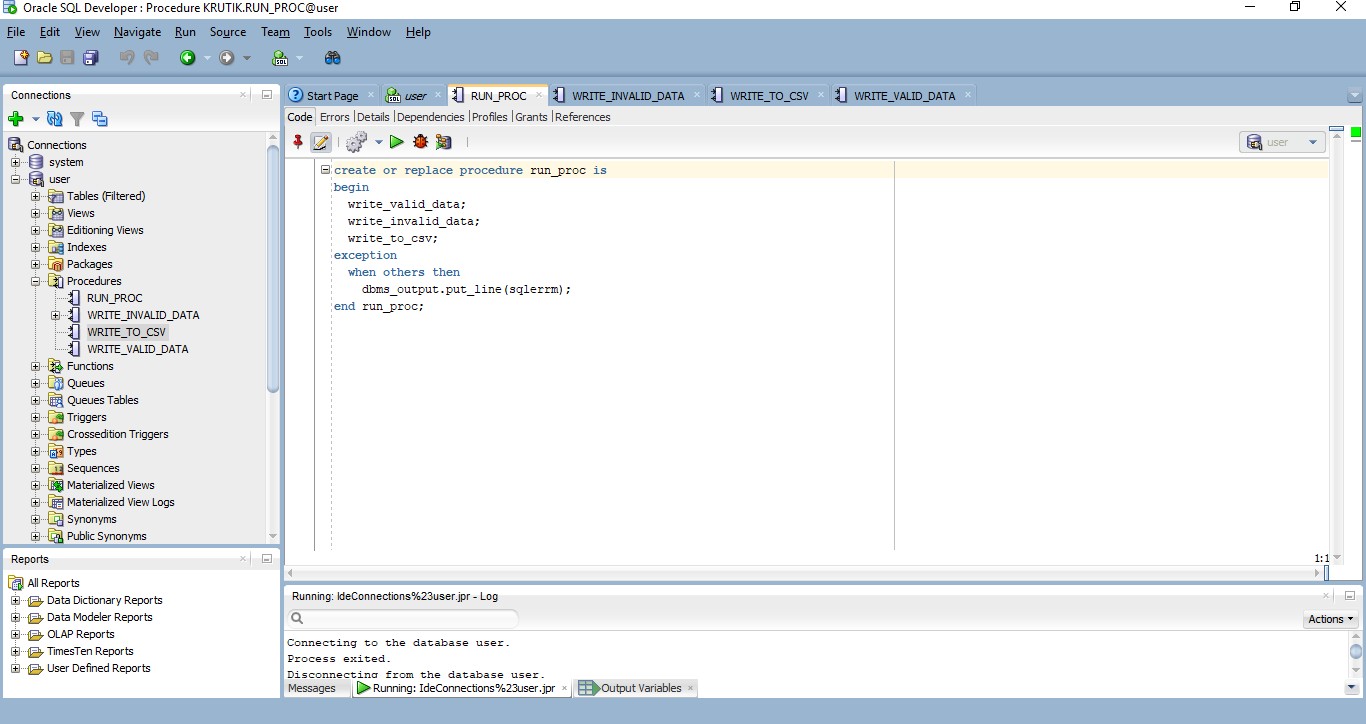
**GRANT READ,WRITE ON** DIRECTORY MF\_KRUTIK\_DIR1 **TO PUBLIC;**

## And grant access to public so we won’t get stuck anywhere in terms of permission

We have write the 3 procedure,

* 1. Will take valid data from donaton\_stg and insert into donation
  2. Will write invalid data from donation\_stg and insert into donation\_err 3- Will make csv file of invalid data (rejected records)

We have used the UTL\_FILE as we are working with OS files that does not allow in put\_line



**create or replace procedure** write\_valid\_data **is begin**

--writing valid data into donation table

**for** i **in (select** dstg**.**address**,**

dstg**.**donorname**,** dstg**.**donationdate**,** dstg**.**donationtime**,** dstg**.**amount**,** dstg**.**donationtype**,** dstg**.**volunteerid**,** ad**.**address\_id

**from** donations\_stg dstg**,** address ad

**where upper(trim(**dstg**.**address**)) = upper(trim(**ad**.**street\_name**)))**

##### loop

**insert into** Donation

**(**Don\_id**,** Donor\_First\_name**,** Donor\_last\_name**,** Donation\_date**,** Donation\_amount**,** Address\_id**,** volunteer\_id**,** donation\_type**)**

##### values

**(**DONATION\_ID**.NEXTVAL,**

**substr(**i**.**donorname**,** 1**, instr(**i**.**donorname**,** ' '**) -** 1**), substr(**i**.**donorname**,**

**instr(**i**.**donorname**,** ' '**) +** 1**, length(**i**.**donorname**)),**

* + 1. donationdate**,** i**.**amount**,**

**(select** ad**.**address\_id

**from** address ad

**where** ad**.**street\_name **=** i**.**address**),** i**.**volunteerid**,**

i**.**donationtype**); end loop;**

##### exception

**when others then**

dbms\_output**.**put\_line**(sqlerrm); end** write\_valid\_data**;**

**create or replace procedure** write\_invalid\_data **is begin**

--writing Invalid data into donation\_err table

**for** i **in (select** dstg**.**address**,**

dstg**.**donorname**,** dstg**.**donationdate**,** dstg**.**donationtime**,** dstg**.**amount**,** dstg**.**donationtype**,** dstg**.**volunteerid

**from** donations\_stg dstg

**where** address **in (select** address

**from** donations\_stg dstg

##### minus

**select** street\_name

**from** address ad**)) loop**

**insert into** donations\_err

**(**Address**,** DonorName**,** DonationDate**,** DonationTime**,** Amount**,** DonationType**,** VolunteerId**)**

##### values

**(**i**.**address**,** i**.**donorname**,** i**.**donationdate**,** i**.**donationtime**,** i**.**amount**,** i**.**donationtype**,** i**.**volunteerid**);**

##### end loop; exception

**when others then**

dbms\_output**.**put\_line**(sqlerrm); end** write\_invalid\_data**;**

**create or replace procedure** write\_to\_csv **is**

--This procedure is used to write the rejected records in CSV file

**CURSOR** cur\_rejected\_records **IS SELECT** d**.**address**,**

d**.**donorname**,** d**.**donationdate**,** d**.**donationtime**,** d**.**amount**,** d**.**donationtype**,** d**.**volunteerid

**FROM** donations\_err d**;**

v\_file UTL\_FILE**.**FILE\_TYPE**; BEGIN**

v\_file **:=** UTL\_FILE**.**FOPEN**(**location **=>** 'MF\_KRUTIK\_DIR1'**,**

filename **=>** 'Rejected\_records.csv'**,** open\_mode **=>** 'W'**,**

max\_linesize **=>** 32767**);**

UTL\_FILE**.**PUT**(**v\_file**,**

'Address' **||** ',' **||** 'Donorname' **||** ',' **||** 'Donationdate' **||**

',' **||**

',' **||**

'Donationtime' **||** ',' **||** 'Amount' **||** ',' **||** 'Donationtype' **||**

'Volunteerid'**);**

**FOR** cur\_rec **IN** cur\_rejected\_records **LOOP**

UTL\_FILE**.**PUT\_LINE**(**v\_file**,**

cur\_rec**.**address **||** ',' **||** cur\_rec**.**donorname **||** ',' **||**

cur\_rec**.**donationdate **||** ',' **||** cur\_rec**.**donationtime **||**

',' **||**

##### END LOOP;

cur\_rec**.**amount **||** ',' **||** cur\_rec**.**donationtype **||** ',' **||**

cur\_rec**.**volunteerid**);**

UTL\_FILE**.**FCLOSE**(**v\_file**);**

##### EXCEPTION

**WHEN OTHERS THEN** UTL\_FILE**.**FCLOSE**(**v\_file**); RAISE;**

**end** write\_to\_csv**;**

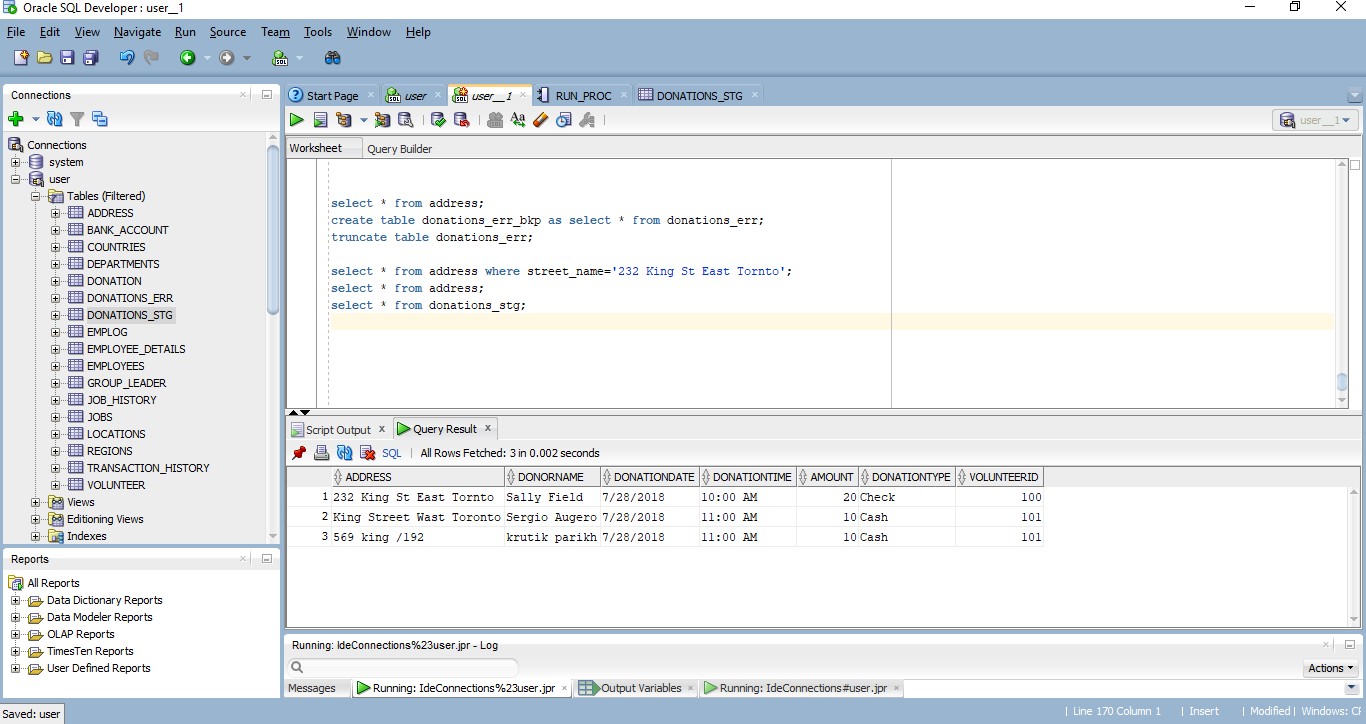
##### create or replace procedure run\_proc is begin

write\_valid\_data**;** write\_invalid\_data**;** write\_to\_csv**;**

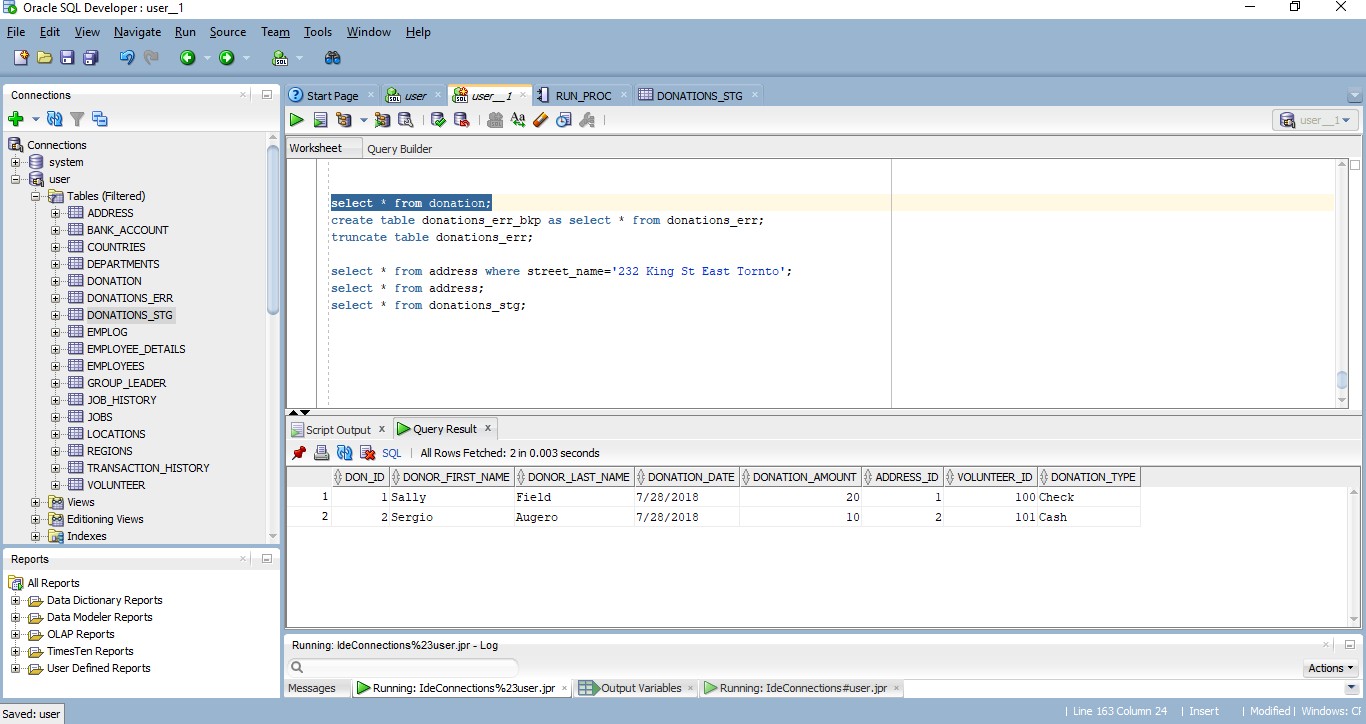
##### exception

**when others then**

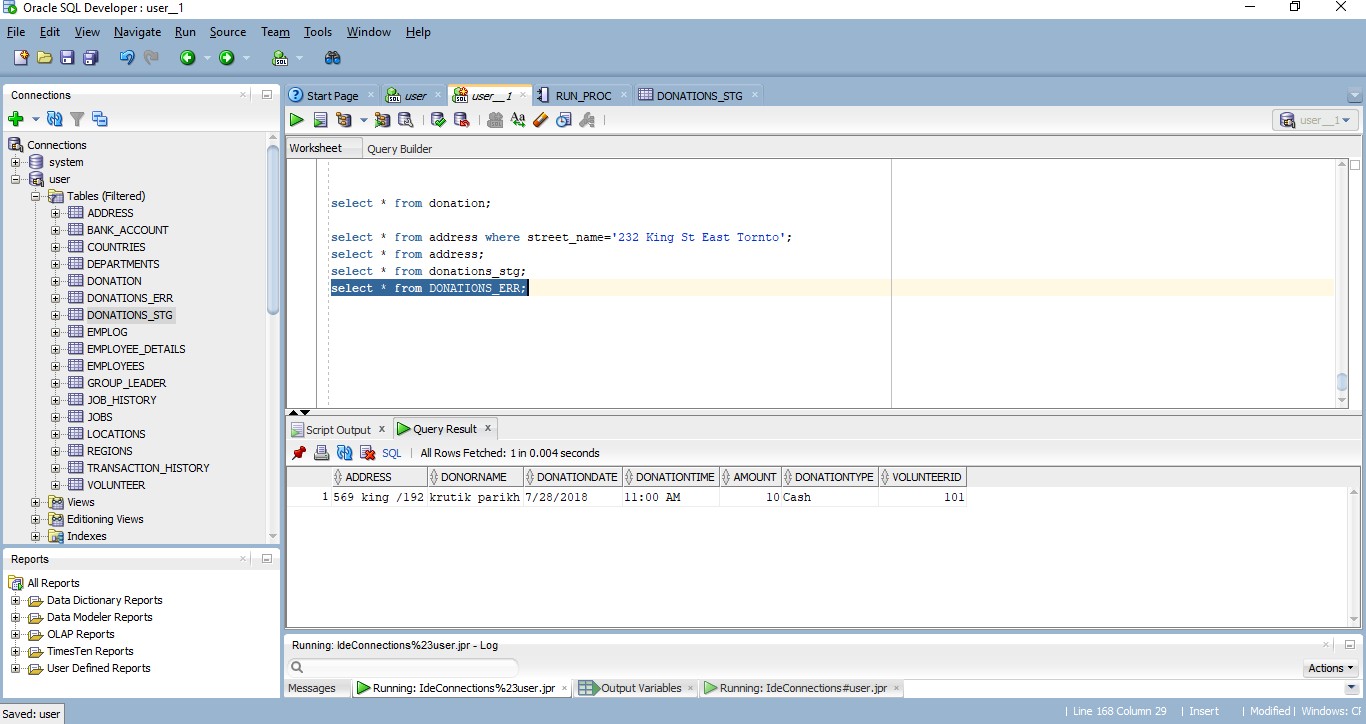
dbms\_output**.**put\_line**(sqlerrm); end** run\_proc**;**

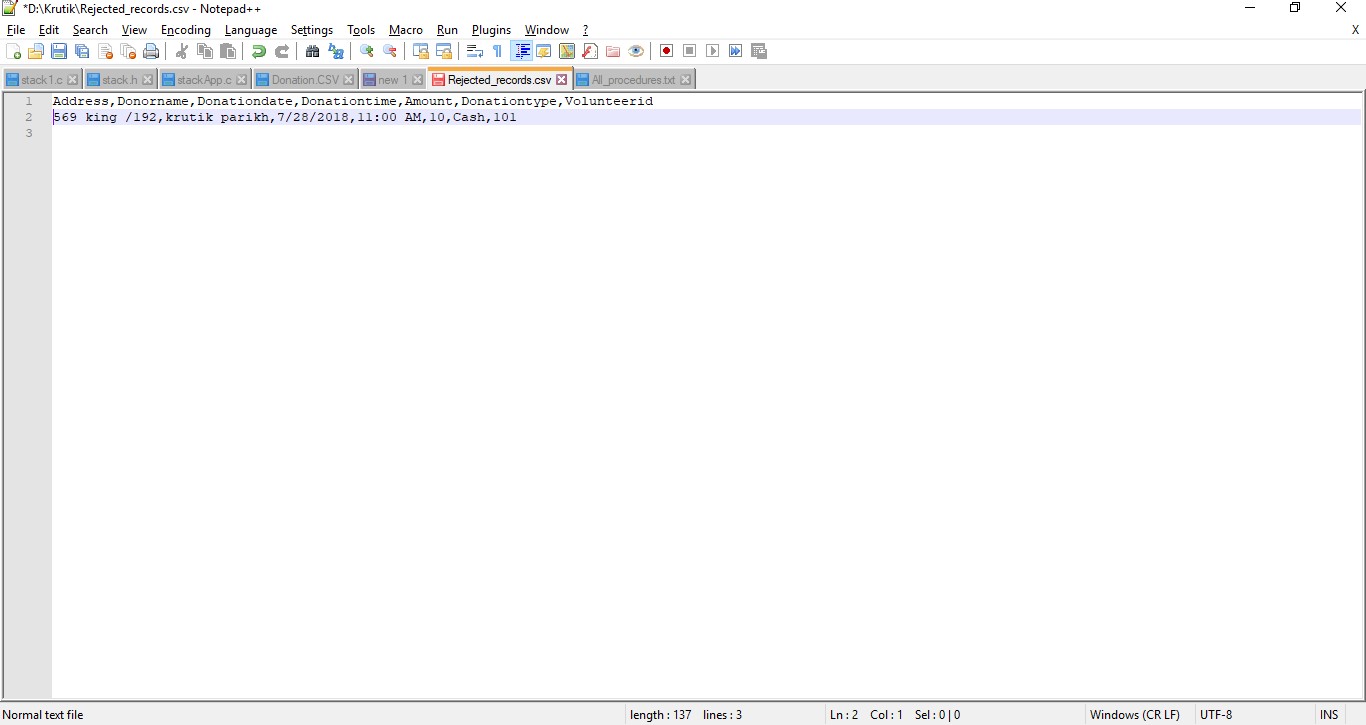
Here is the database donation.stg which stored the correct and incorrect address of donations.txt file

After running 3 procedure lets check whether the correct and incorrect data is stored in database or not. Here is the database donation which stores the correct address



Here is the database donation.err which stores the incorrect or invalid address of donation.txt.



Lets see where the database make csv file for rejected records

So here is the reject\_records which have entered by us when we are loading data from donation.txt

# Task 4:

## We have entered some data manually as Sheridan database has no database table except address.

**create or replace view** view\_day

##### as

**select** donation\_date**,sum(**donation\_amount**)** Total\_amt**,avg(**donation\_amount**)**

Avg\_amt **from** donation

**group by** donation\_date**;**

**create or replace view** view\_month

##### as select

**decode(substr(**donation\_date**,**1**,**2**),**'01'**,**'JAN'**,**'1'**,**'JAN'**,**'02'**,**'FEB'**,**'2'**,**'FEB'**,**'0 3'**,**'MAR'**,**'3'**,**'MAR'**,**

'04'**,**'APR'**,**'4'**,**'APR'**,**

'05'**,**'MAY'**,**'5'**,**'MAY'**,**

'06'**,**'JUN'**,**'6'**,**'JUN'**,**

'07'**,**'JUL'**,**'7'**,**'JUL'**,**

'08'**,**'AUG'**,**'8'**,**'AUG'**,**

'09'**,**'SEP'**,**'9'**,**'SEP'**,** '10'**,**'OCT'**,**

'11'**,**'NOV'**,**

'12'**,**'DEC'

**) MONTH,sum(**donation\_amount**)** Total\_amt**,AVG(**donation\_amount**)** avg\_amt **from**

donation

**group by substr(**donation\_date**,**1**,**2**);**

**create or replace view** view\_by\_address **as select** ad**.**street\_name**,sum(**d**.**donation\_amount**)**

Total\_amount**,avg(**d**.**donation\_amount**)** Avg\_amount **from** address ad**,**donation d

**where** ad**.**address\_id**=**d**.**address\_id

**group by** ad**.**street\_name**;**

**create or replace view** view\_by\_postal\_code **as select** ad**.**postal\_code**,sum(**d**.**donation\_amount**)**

Total\_amount**,avg(**d**.**donation\_amount**)** Avg\_amount **from** address ad**,**donation d

**where** ad**.**address\_id**=**d**.**address\_id

**group by** ad**.**postal\_code**;**

**create or replace view** view\_by\_volunteer **as**

**select** volunteer\_id**,sum(**donation\_amount**)** Total\_amt**,avg(**donation\_amount**)**

Avg\_amt **from** donation d

**group by** volunteer\_id**;**

**create or replace view** view\_by\_group\_lead **as select** gl**.**GROUP\_LEAD\_ID**,sum(**d**.**donation\_amount**)**

Total\_amount**,avg(**d**.**donation\_amount**)** Avg\_amount **from** group\_leader gl**,**donation d

**where** gl**.**volunteer\_id**=**d**.**volunteer\_id

**group by** gl**.**GROUP\_LEAD\_ID**;**

# Task 5:

So I task 6 we are using oracle 12c and it does not allow user to create dmluser so instead of that we have use c##dmluser after getting errors

[**https://stackoverflow.com/questions/33330968/error-ora-65096-invalid-common-user-or-role-name-**](https://stackoverflow.com/questions/33330968/error-ora-65096-invalid-common-user-or-role-name-in-oracle)[**in-oracle**](https://stackoverflow.com/questions/33330968/error-ora-65096-invalid-common-user-or-role-name-in-oracle)

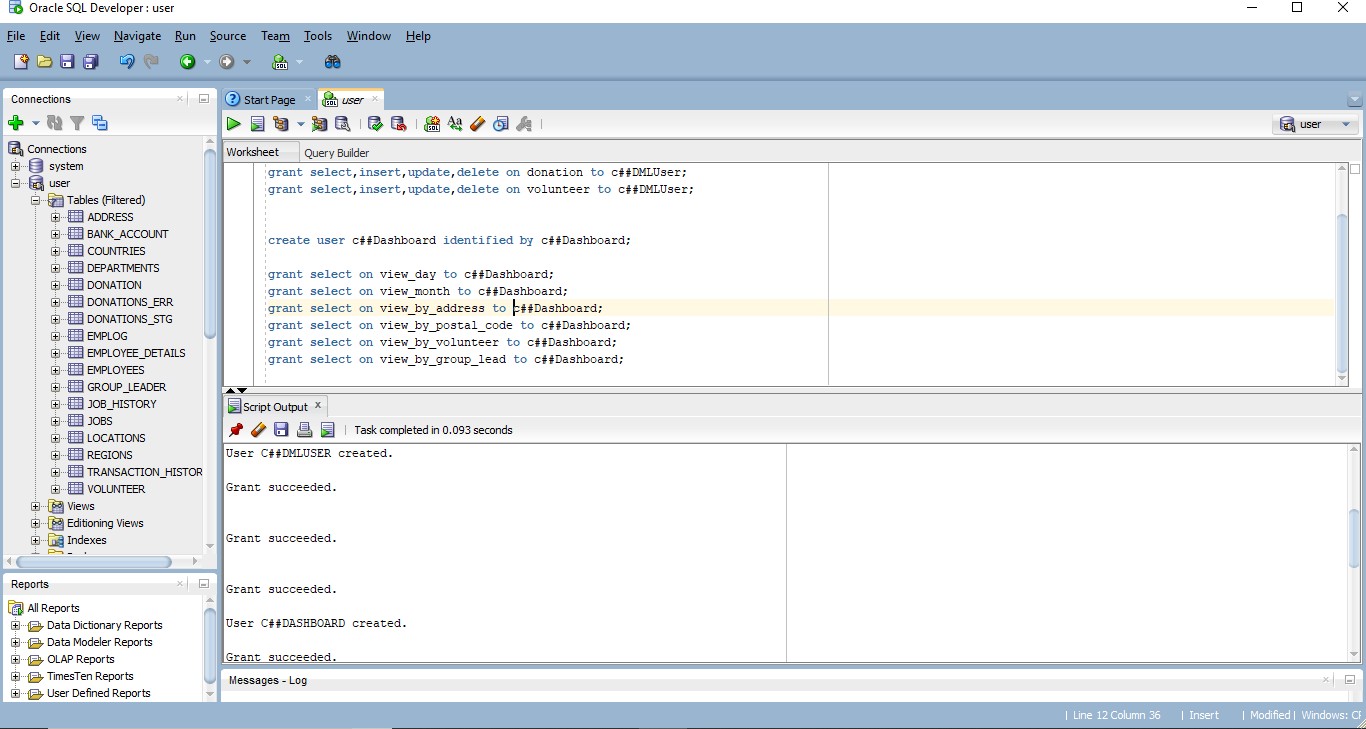
**create user** c##DMLUser **identified by** c##DMLUser**;**

**grant select,insert,update,delete on** address **to** c##DMLUser**; grant select,insert,update,delete on** donation **to** c##DMLUser**; grant select,insert,update,delete on** volunteer **to** c##DMLUser**;**

**create user** c##Dashboard **identified by** c##Dashboard**; grant select on** view\_day **to** c##Dashboard**;**

**grant select on** view\_month **to** c##Dashboard**; grant select on** view\_by\_address **to** c##Dashboard**;**

**grant select on** view\_by\_postal\_code **to** c##Dashboard**; grant select on** view\_by\_volunteer **to** c##Dashboard**; grant select on** view\_by\_group\_lead **to** c##Dashboard**;**



Queries for how we have created the tables and the values we have entered into table.

--Drop table

**drop table** ADDRESS**;**

-- Create table **create table** ADDRESS **(**

address\_id **NUMBER null,**

unit\_num **VARCHAR2(**6**),** street\_number **NUMBER not null,** street\_name **VARCHAR2(**24**) not null,** street\_type **VARCHAR2(**12**) null,** street\_description **CHAR(**1**),**

postal\_code **CHAR(**7**) null,**

city **VARCHAR2(**16**) not null,**

province **CHAR(**10**) not null**

**);**

-- Create/Recreate primary, unique and foreign key constraints

**alter table** ADDRESS

**add primary key (**ADDRESS\_ID**);**

-- Create/Recreate check constraints

**alter table** ADDRESS

**add constraint** ST\_DIR

**check (**Street\_description **in(**'E'**,**'W'**,**'N'**,**'S'**));**

-- Grant/Revoke object privileges

**grant select, insert, update, delete on** ADDRESS **to** c##DMLUSER**;**

-- Create table

**drop table** GROUP\_LEADER**; create table** GROUP\_LEADER

**(**

volunteer\_id **NUMBER not null,** first\_name **VARCHAR2(**16**),** last\_name **VARCHAR2(**16**),** group\_lead\_id **NUMBER**

**);**

-- Create/Recreate primary, unique and foreign key constraints

**alter table** GROUP\_LEADER

**add primary key (**VOLUNTEER\_ID**);**

--Insert script

**insert into** GROUP\_LEADER **(**VOLUNTEER\_ID**,** FIRST\_NAME**,** LAST\_NAME**,** GROUP\_LEAD\_ID**)**

**values (**100**,** 'Michael'**,** 'Clarke'**,** 100**);**

**insert into** GROUP\_LEADER **(**VOLUNTEER\_ID**,** FIRST\_NAME**,** LAST\_NAME**,** GROUP\_LEAD\_ID**)**

**values (**101**,** 'Allen'**,** 'Border'**,** 100**);**

**drop table** VOLUNTEER**;**

-- Create table

**create table** VOLUNTEER

**(**

volunteer\_id **NUMBER not null,**

first\_name **VARCHAR2(**16**),** last\_name **VARCHAR2(**16**),** group\_leader **VARCHAR2(**16**)**

**);**

-- Create/Recreate primary, unique and foreign key constraints

**alter table** VOLUNTEER

**add primary key (**VOLUNTEER\_ID**); alter table** VOLUNTEER

**add constraint** FK\_VOL\_LEAD **foreign key (**VOLUNTEER\_ID**) references** GROUP\_LEADER **(**VOLUNTEER\_ID**);**

-- Grant/Revoke object privileges

**grant select, insert, update, delete on** VOLUNTEER **to** c##DMLUSER**;**

--insert script

**insert into** VOLUNTEER **(**VOLUNTEER\_ID**,** FIRST\_NAME**,** LAST\_NAME**,** GROUP\_LEADER**)**

**values (**100**,** 'Mark'**,** 'Waugh'**,** '100'**);**

**insert into** VOLUNTEER **(**VOLUNTEER\_ID**,** FIRST\_NAME**,** LAST\_NAME**,** GROUP\_LEADER**)**

**values (**101**,** 'Steve'**,** 'Waugh'**,** '101'**);**

**drop table** DONATION**;**

-- Create table **create table** DONATION **(**

don\_id **NUMBER not null,** donor\_first\_name **VARCHAR2(**16**),** donor\_last\_name **VARCHAR2(**16**),** donation\_date **VARCHAR2(**20**) not null,** donation\_amount **NUMBER not null,** address\_id **NUMBER not null,** volunteer\_id **NUMBER not null,** donation\_type **VARCHAR2(**16**)**

**);**

-- Create/Recreate primary, unique and foreign key constraints

**alter table** DONATION

**add primary key (**DON\_ID**); alter table** DONATION

**add constraint** FK\_DON\_ADD **foreign key (**ADDRESS\_ID**) references** ADDRESS **(**ADDRESS\_ID**);**

**alter table** DONATION

**add constraint** FK\_DON\_VOL **foreign key (**VOLUNTEER\_ID**) references** VOLUNTEER **(**VOLUNTEER\_ID**);**

-- Grant/Revoke object privileges

**grant select, insert, update, delete on** DONATION **to** c##DMLUSER**;**

-- Create directory

**create or replace** directory MF\_KRUTIK\_DIR1

**as** 'D:\Krutik'**;**

**GRANT READ,WRITE ON** DIRECTORY MF\_KRUTIK\_DIR1 **TO PUBLIC;**

**select \* from** donations\_stg**;**

**drop table** donations\_stg**;**

**CREATE TABLE** donations\_stg

**(**

Address **varchar2(**50**),** DonorName **varchar2(**50**),** DonationDate **varchar2(**50**),** DonationTime **varchar2(**50**),** Amount **number,** DonationType **varchar2(**20**),** VolunteerId **number**

**)**

**CREATE TABLE** donations\_err

**(**

Address **varchar2(**50**),** DonorName **varchar2(**50**),** DonationDate **varchar2(**50**),** DonationTime **varchar2(**50**),** Amount **number,** DonationType **varchar2(**20**),** VolunteerId **number**

**);**