**Trigger to check if the driver and vehicle are available for that period of time**

create trigger available before insert on booking for each row execute procedure checkBooking();

set search\_path to car\_rental;

create or replace function checkBooking() returns trigger as $body$

DECLARE

d Driver%rowtype;

v vehicle%rowtype;

bk Booking%rowtype;

assigned driver.d\_id%type;

give boolean;

flag boolean;

BEGIN

flag='yes';

assigned=NEW.d\_ID;

if(tg\_op='INSERT') then

if(NEW.start\_time>NEW.end\_time)then

return NULL;

end if;

if(NEW.Driver\_Requirement='yes') then

For bk in select \* from booking

Loop

if(bk.Driver\_Requirement='yes' AND bk.d\_ID=NEW.d\_ID) then

if(NEW.start\_time<bk.end\_time) then

flag='no';

end if;

end if;

End Loop;

if(flag='no')then

For d in select \* from driver

Loop

if(d.d\_id!=NEW.d\_id)then

give='yes';

for bk in select \* from booking

Loop

if(bk.d\_id=d.d\_id)then

if(bk.end\_time>NEW.start\_time)then

give='no';

end if;

end if;

End Loop;

if(give='yes')then

assigned=d.d\_id;

EXIT;

end if;

end if;

End Loop;

if(assigned!=NEW.d\_id)then

NEW.d\_id=assigned;

else

return NULL;

end if;

end if;

else

NEW.d\_id=NULL;

end if;

For v in select \* from Vehicle

LOOP

For bk in select \* from booking

LOOP

if(bk.Vehicle\_ID=NEW.Vehicle\_ID) then

if(NEW.start\_time<bk.end\_time) then

return NULL;

end if;

end if;

End Loop;

End Loop;

return NEW;

end if;

END;

$body$ language 'plpgsql'

**Trigger to check if the user is giving proper valid ratings or not**

create trigger CheckRatings before insert on Billing for each row execute procedure checkRating();

set search\_path to car\_rental;

create or replace function checkRating() returns trigger as $body$

DECLARE

bk booking%rowtype;

BEGIN

for bk in select \* from booking

LOOP

if(NEW.Book\_ID=bk.Book\_ID)then

if(bk.Driver\_Requirement='yes')then

if(NEW.Driver\_Rating <1 or NEW.Driver\_Rating>5)then

return NULL;

else

if(NEW.veh\_Rating <1 or NEW.veh\_Rating>5)then

return NULL;

else

return NEW;

end if;

end if;

else

NEW.Driver\_Rating=0;

if(NEW.veh\_Rating <1 or NEW.veh\_Rating>5)then

return NULL;

else

return NEW;

end if;

end if;

end if;

END LOOP;

END;

$body$ language 'plpgsql'

**Trigger to find the total amount in the bill for each tuple at the time of insertion**

create trigger calculate before insert on Billing for each row execute procedure calculateFinal();

set search\_path to car\_rental;

create or replace function calculateFinal() returns trigger as $body$

DECLARE

d Driver%rowtype;

v Vehicle%rowtype;

a d\_allowances%rowtype;

bk booking%rowtype;

totalday real;

totalmile real;

mybook booking%rowtype;

myveh vehicle%rowtype;

driver driver%rowtype;

allow d\_allowances%rowtype;

tt real;

total real;

BEGIN

tt:=0;

for bk in select \* from booking

loop

if(bk.book\_id=NEW.book\_id)then

mybook=bk;

end if;

end loop;

if(mybook.driver\_requirement='no')then

NEW.driver\_rating=NULL;

end if;

for v in select \* from vehicle

loop

if(mybook.vehicle\_id=v.vehicle\_id)then

myveh=v;

end if;

end loop;

if(mybook.driver\_requirement='yes')then

for d in select \* from driver

loop

if(mybook.d\_id=d.d\_id)then

driver=d;

end if;

end loop;

for a in select \* from d\_allowances

loop

if(driver.a\_id=a.a\_id)then

allow=a;

end if;

end loop;

tt:=tt+(mybook.end\_time-mybook.start\_time+1)\*(allow.amount);

end if;

totalmile:=(NEW.f\_reading-NEW.init\_reading)\*(myveh.cost\_per\_mile)+tt;

totalday:=(mybook.end\_time-mybook.start\_time+1)\*(myveh.cost\_per\_day)+tt;

if(totalmile>totalday)then

total=totalmile;

else

total=totalday;

end if;

NEW.tax\_amount=total\*(0.18);

NEW.total\_amount=total+NEW.tax\_amount;

return NEW;

END;

$body$ language 'plpgsql'

**Source Code for Console Application**

#include <stdio.h>

#include <stdlib.h>

#include "libpq-fe.h"

PGresult \*res;

void do\_exit(PGconn \*conn) {

PQfinish(conn);

exit(1);

}

void ExecuteQuery(PGconn\* conn,char\* query){

PQexec(conn, "set search\_path to car\_rental;");

res = PQexec(conn, query);

if (PQresultStatus(res) != PGRES\_TUPLES\_OK) {

fprintf(stderr, "Query failed: %s",PQerrorMessage(conn));

do\_exit(conn);

}

else{

int rows = PQntuples(res);

int ncols = PQnfields(res);

for (int i=0; i<ncols; i++){

char \*name = PQfname(res, i);

printf("%s ", name);

}

printf("\n");

for(int i=0; i<rows; i++) {

for(int j=0;j<ncols;j++){

printf("%s ",PQgetvalue(res, i, j));

}

printf("\n");

}

}

}

void ExecuteUpdate(PGconn\* conn,char\* command){

PQexec(conn, "set search\_path to car\_rental;");

res = PQexec(conn, command);

if (PQresultStatus(res) != PGRES\_COMMAND\_OK){

fprintf(stderr, "%.6s failed: %s",command,PQerrorMessage(conn));

do\_exit(conn);

}

else{

printf("successful\n");

}

}

int main()

{

const char \*conninfo = "dbname=postgres user=postgres password=qwerty12345 hostaddr=127.0.0.1 port=5432";

PGconn \*conn = PQconnectdb(conninfo);

if (PQstatus(conn) != CONNECTION\_OK){

fprintf(stderr, "Connection to database failed: %s\n",PQerrorMessage(conn));

do\_exit(conn);

}

char \*user = PQuser(conn);

char \*db\_name = PQdb(conn);

printf("User: %s\n", user);

printf("Database name: %s\n", db\_name);

while(1){

char command[1000];

printf("Enter Command in a single line\n");

fgets (command, 1000, stdin);

if(command[0]=='S' || command[0]=='s'){

ExecuteQuery(conn,command);

}

else{

ExecuteUpdate(conn,command);

}

PQclear(res);

}

PQfinish(conn);

return 0;

}