CSI 2132 Winter 2019
Instructor: Verena Kantere

Hotel Project Report

Nicholas Allair 8147249
Saif Zabarah 2359049

DBMS and Programing Language

For our implementation of the hotel chain application, we used PostgreSQL as a database management system. Postgres allows for future growth of the project and standard compliance. Postgres allows for small single table applications, to large web-page applications which may face many tables, triggers, and multiple, concurrent users. PgAdmin 4 allowed us to create a local host for the back end support of our web app, it also allows us to run queries and insertions to our database. Our user interface was programmed in PHP, which is exceptionally suited for fast and flexible web development. PHP scripts allow us to easily connect the web application to the database using the host, database name and user info. Furthermore, our webpage was styled using css.

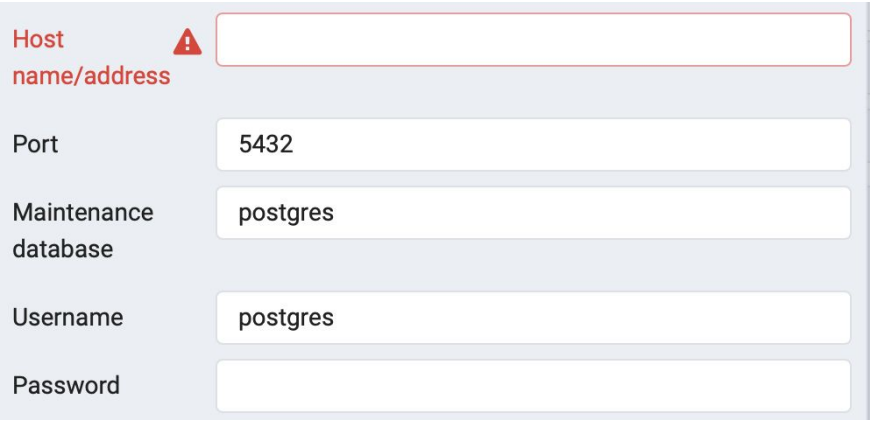
Upon launching the application the user will view a home page which gives information about which link they should choose. An existing customer can sign in using their email address and password , where they can book a room, and view their existing booking(s). The employee tab allows a existing employee to log in using their SSN number and password. Here they can create a booking for a customer. The admin tab allows for the creation, edit, and removal of any aspect of the database, from a hotel chain to the room of a specific hotel.

Steps for Installation

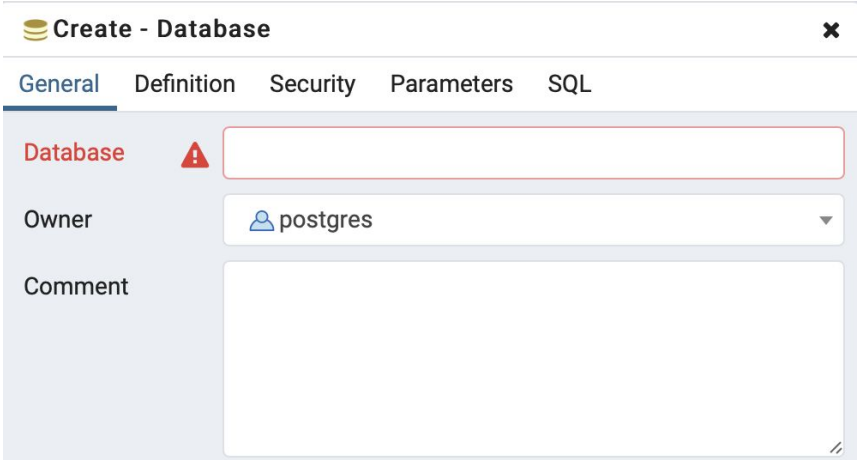
In order to setup the application for use, the following steps must be followed.

1. If not already installed, download a GUI for postgres queries. Ex. pgAdmin.

2. Setup a server, it is important to take note of the host name, port id, username, and password of your server. This information will be required in order to link the web application to your DB. For a local database, the host should be set to localhost, and the port can be left as the default.



3. Once your server is setup, a database must be created. The name of your database will be needed in order to link the DB to the webapp.



4. Using the DDL's found in DataDefinition.sql, all tables triggers and functions of the database can be built.

5. The database can then be populated, with the data specified in the project description, py running the sql code found in insertions-2.sql.

6. Once your database is setup, it is time to link it with the webapp. In order to do so, the information in function __construct() , which is found in the source code db_connection.php, must be changed to the information which corresponds to the database that you created earlier.

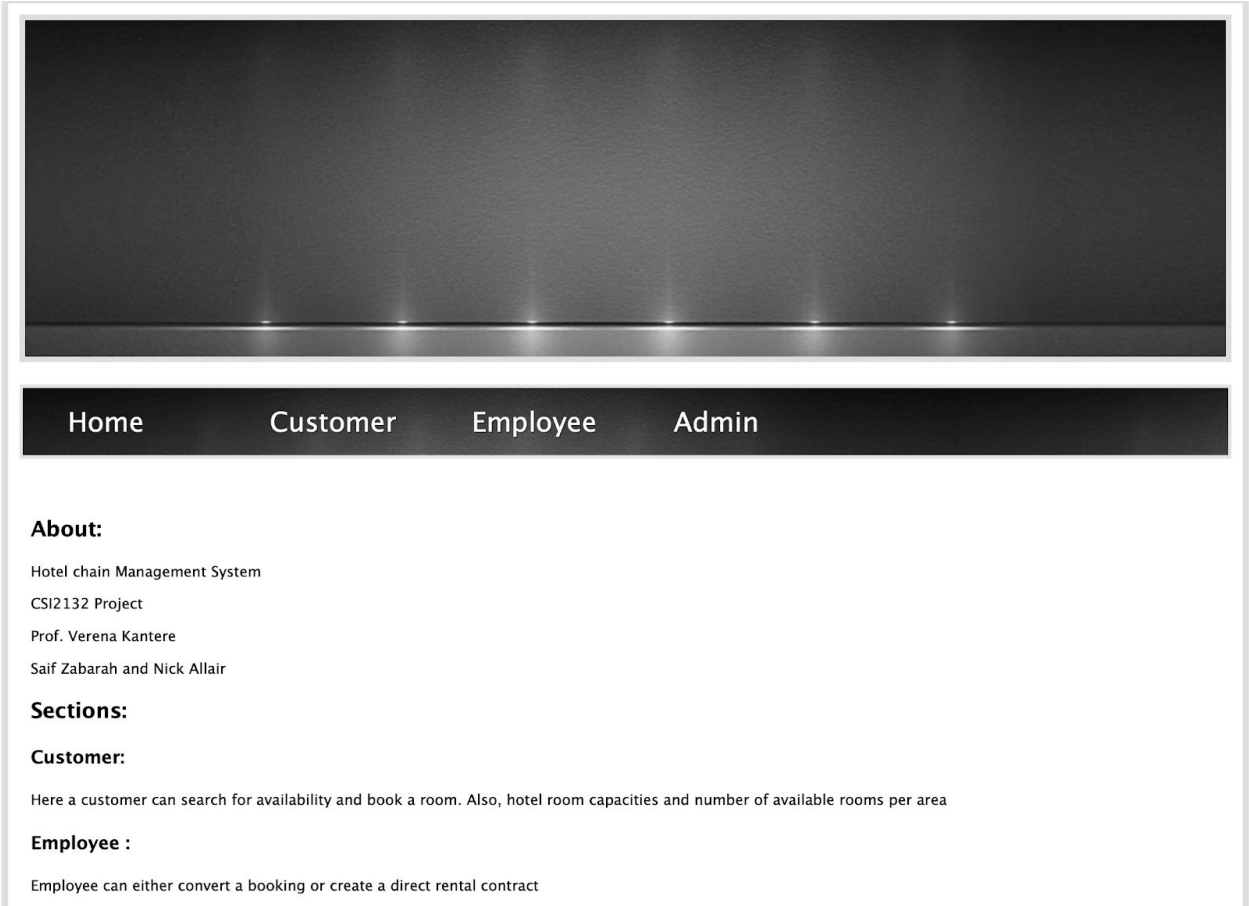
```
function __construct() {

    $this->host = "host = localhost";
    $this->port = "port = 5432";
    $this->dbname = "dbname = project_hotel_db";
    $this->credentials = "user = postgres password=Hello123";
```

The host, port, database name, username and password must correspond to what you have data you have entered earlier in setup.

7. You are now ready to launch the php web application. In order to launch the php web app, proprietary to create a local server. Mamp(for mac os) or Wamp(for windows) is an easy to use software which can create this local server. And allow us to run scripts easily. Other apache software can also be used.

8. Once your web start page is has been opened, navigate to: http://localhost:8888/project_hotel/Sites/index.php (your local host numbers may vary) This will link to the index.php file, and the homepage for the web app.



9. You can test to see if the connection to your database has been made by trying to access any of the admin function. If the connection is not successful, an error message will be displayed at the top of the screen.

10. You can now create a profile under the admin menu, and book rooms. Furthermore, you can sign in as one of the pre-existing employees created from insertions, or create a new employee

DDL List

1.Table Creations

```
CREATE TABLE hotel_chain (
   name character varying(500),
   central_office_address_street_num integer NOT NULL,
central_office_address_street_name character varying(500),
central_office_address_city character varying(500),
central_office_address_prov character varying(500),
central_office_address_postal character varying(500),
number_of_hotels integer DEFAULT 0,
        PRIMARY KEY (name)
);


CREATE TABLE hotel (
hotel_ID integer NOT NULL,
name character varying(500),
chain character varying(500),
hotel_address_Street_num integer NOT NULL,
address_street_name character varying(500),
address_city character varying(500),
address_prov character varying(500),
address_postal character varying(500),
number_of_rooms INTEGER DEFAULT 0, /*derived */
rating numeric(5,1),
/*manager,*/
gym boolean,
pool boolean,
        PRIMARY KEY (hotel_ID),
        Foreign KEY (chain)  References hotel_chain
                ON UPDATE CASCADE ON DELETE CASCADE
);
```

```
CREATE TABLE Chain_emails(
chain_name character varying(500),
email character varying(500),
constraint chk_email check (email like '%_@__%.__%'),
PRIMARY KEY (email, chain_name) ,
Foreign KEY (chain_name)  References hotel_chain
                ON UPDATE CASCADE ON DELETE CASCADE
);


CREATE TABLE Chain_phone_num(
chain_name character varying(500),
number char(10),
CONSTRAINT chk_phone CHECK (number not like '%[^0-9]%'),
PRIMARY KEY (number, chain_name),
Foreign KEY (chain_name)  References hotel_chain
                ON UPDATE CASCADE ON DELETE CASCADE
);


CREATE TABLE Hotel_emails(
hotel_id integer NOT NULL,
email character varying(500),
constraint chk_email check (email like '%_@__%.__%'),
PRIMARY KEY (email, hotel_id) ,
Foreign KEY (hotel_id)  References hotel
                ON UPDATE CASCADE ON DELETE CASCADE
);


CREATE TABLE Hotel_phone_num(
hotel_id integer NOT NULL,
number char(10),
CONSTRAINT chk_phone CHECK (number not like '%[^0-9]%'),
PRIMARY KEY (hotel_id, number),
Foreign KEY (hotel_id)  References hotel
                ON UPDATE CASCADE ON DELETE CASCADE
);
```

```
CREATE TABLE Room(
hotel_ID integer NOT NULL,
room_num integer NOT NULL,
capacity integer NOT NULL,
damages character varying(500),
price MONEY,
tv boolean,
airconditioning boolean,
fridge boolean,
view_ character varying(500),
possible_extend boolean,
PRIMARY KEY (hotel_ID, room_num),
Foreign KEY (hotel_ID)  References hotel
                    ON UPDATE CASCADE ON DELETE CASCADE
);


CREATE TABLE customer(
email character varying(500),
constraint chk_email check (email like '%_@__%.__%'),
first_name character varying(500),
last_name character varying(500),
phone_number char(10),
CONSTRAINT chk_phone CHECK (phone_number not like '%[^0-9]%'),
address_street_num integer NOT NULL,
address_street_name character varying(500),
address_city character varying(500),
address_prov character varying(500),
address_postal character varying(500),
date_reg DATE NOT NULL DEFAULT(NOW()),
password character varying(500),
PRIMARY KEY (email)
);
```

```
CREATE TABLE Employee(
SSN_SIN char(9),
CONSTRAINT chk_ssn CHECK (SSN_SIN not like '%[^0-9]%'),
hotel integer NOT NULL,
email character varying(500),
constraint chk_email check (email like '%_@__%.__%'),
first_name character varying(500),
last_name character varying(500),
phone_num char(10),
CONSTRAINT chk_phone CHECK (phone_num not like '%[^0-9]%'),
address_Street_num integer NOT NULL,
address_street_name character varying(500),
address_city character varying(500),
address_prov character varying(500),
address_postal character varying(500),
status character varying(500),
password character varying(500),
PRIMARY KEY (SSN_SIN),
Foreign KEY (hotel)  References hotel
                ON UPDATE CASCADE ON DELETE CASCADE

);


ALTER TABLE hotel ADD manager char(9) ;
ALTER TABLE hotel ADD CONSTRAINT chk_ssn CHECK (manager not like '%[^0-9]%');
ALTER TABLE hotel ADD Foreign KEY (manager)  References Employee
                ON UPDATE RESTRICT ON DELETE RESTRICT
;


CREATE TABLE Employee_roles(
employee_id  char(9),
CONSTRAINT chk_ssn CHECK (employee_id not like '%[^0-9]%'),
role character varying(500),
PRIMARY KEY(employee_id, role),
Foreign KEY (employee_id)  References Employee
                ON UPDATE CASCADE ON DELETE CASCADE
```

```
);
CREATE TABLE Supervise(
supervisor char(9),
supervisee char(9),
CONSTRAINT chk_ssn CHECK (supervisor not like '%[^0-9]%'),
CONSTRAINT chk_ssn2 CHECK (supervisee not like '%[^0-9]%'),
PRIMARY KEY(supervisor, supervisee),
Foreign KEY (supervisor)  References Employee
                ON UPDATE CASCADE ON DELETE CASCADE,
Foreign KEY (supervisee)  References Employee
                ON UPDATE CASCADE ON DELETE CASCADE
);


CREATE TABLE Booking(
booking_id Integer GENERATED BY DEFAULT AS IDENTITY NOT NULL,
customer character varying(500),
constraint chk_email check (customer like '%_@__%.__%'),
hotel Integer NOT NULL,
room Integer NOT NULL,
check_In_Date DATE,
check_Out_Date DATE,
cancelled boolean,
notes character varying(500),
PRIMARY KEY(booking_id),
Foreign KEY (customer)  References customer
                ON UPDATE CASCADE ON DELETE CASCADE,
Foreign KEY (hotel, room)  References Room
                ON UPDATE CASCADE ON DELETE CASCADE
);
```

```
CREATE TABLE Rental_contract(
rental_ID Integer GENERATED BY DEFAULT AS IDENTITY NOT NULL,
hotel Integer NOT NULL,
room Integer NOT NULL,
customer character varying(500),
constraint chk_email check (customer like '%_@__%.__%'),
check_in_empl char(9),
CONSTRAINT chk_ssn CHECK (check_in_empl not like '%[^0-9]%'),
check_In_Date Date,
check_Out_Date Date,
notes character varying(500),
payment_total MONEY DEFAULT 0,
total_cost MONEY,
balance MONEY, /*trigger*/
PRIMARY KEY(rental_ID),
Foreign KEY (customer)  References customer
                ON UPDATE CASCADE ON DELETE CASCADE,
Foreign KEY (hotel, room)  References Room
                ON UPDATE CASCADE ON DELETE CASCADE,
Foreign KEY (check_in_empl)  References Employee
                ON UPDATE CASCADE ON DELETE CASCADE
);


CREATE TABLE Payment(
Payment_ID Integer GENERATED BY DEFAULT AS IDENTITY not NULL,
rental_contract Integer not NULL,
pay_date DATE,
pay_type character varying(500),
amount_Charged MONEY,
PRIMARY KEY(Payment_ID),
Foreign KEY (rental_contract)  References Rental_contract
                ON UPDATE CASCADE ON DELETE CASCADE
);
```

```
CREATE TABLE Convert_booking(
Booking Integer not NULL,
rental_agreement Integer not NULL,
PRIMARY KEY(Booking, rental_agreement),
Foreign KEY (Booking)  References Booking
                ON UPDATE CASCADE ON DELETE CASCADE,
Foreign KEY (rental_agreement)  References Rental_contract
                ON UPDATE CASCADE ON DELETE CASCADE
);
```

```
CREATE TABLE Archive(
archive_ID Integer GENERATED BY DEFAULT AS IDENTITY not NULL,
check_In_Date Date,
check_Out_Date Date,
PRIMARY KEY(archive_ID)
);
```

```
CREATE TABLE archive_room(
archive_id Integer not NULL,
hotel Integer not NULL,
room Integer not NULL,
PRIMARY KEY(archive_ID),
Foreign KEY (archive_id)  References archive
                ON UPDATE CASCADE ON DELETE CASCADE,
Foreign KEY (hotel, room)  References Room
                ON UPDATE CASCADE ON DELETE CASCADE
);
```

```
CREATE TABLE Archive_checkin(
archive_id Integer not NULL,
check_in_empl char(9),
CONSTRAINT chk_ssn CHECK (check_in_empl not like '%[^0-9]%'),
PRIMARY KEY(archive_id),
Foreign KEY (archive_id)  References archive
                ON UPDATE CASCADE ON DELETE CASCADE,
Foreign KEY (check_in_empl)  References Employee
                ON UPDATE CASCADE ON DELETE CASCADE
);


CREATE TABLE Archive_contract(
archive_id Integer not NULL,
contract_id Integer not NULL,
PRIMARY KEY(archive_id),
Foreign KEY (archive_id)  References archive
                ON UPDATE CASCADE ON DELETE CASCADE,
Foreign KEY (contract_id)  References Rental_contract
                ON UPDATE CASCADE ON DELETE CASCADE
);


CREATE TABLE Archive_booking(
archive_id Integer not NULL,
booking_id Integer not NULL,
PRIMARY KEY(archive_id),
Foreign KEY (archive_id)  References archive
                ON UPDATE CASCADE ON DELETE CASCADE,
Foreign KEY (booking_id)  References Booking
                ON UPDATE CASCADE ON DELETE CASCADE
);
```

```
CREATE TABLE Archive_customer(
archive_id Integer not NULL,
customer_email character varying(500),
constraint chk_email check (customer_email like '%_@__%.__%'),
PRIMARY KEY(archive_id),
Foreign KEY (archive_id)  References archive
                ON UPDATE CASCADE ON DELETE CASCADE,
Foreign KEY (customer_email)  References customer
                ON UPDATE CASCADE ON DELETE CASCADE
);
```

2. Trigger Creation

```
create trigger hotel_updates_trig
after insert on hotel
for each row
EXECUTE PROCEDURE hotels_num();

create trigger balance_updates_trig
before insert or update on rental_contract
for each row
EXECUTE PROCEDURE balance_update();

create trigger payment_updates_trig
after insert on payment
for each row
EXECUTE PROCEDURE payment_update();

create trigger incorrect_dates
after insert or update on booking
for each row
when (NEW.check_in_date > NEW.check_out_date or NEW.check_in_date <
CURRENT_DATE or NEW.check_in_date < CURRENT_DATE)
EXECUTE PROCEDURE rollback_entry();
```

```
create trigger incorrect_dates
after insert or update on rental_contract
for each row
when (NEW.check_in_date > NEW.check_out_date or NEW.check_in_date <
CURRENT_DATE or NEW.check_in_date < CURRENT_DATE)
EXECUTE PROCEDURE rollback_entry();

create trigger archive_booking
after insert or update on booking
for each row
EXECUTE PROCEDURE arch_book_func();

create trigger archive_rental
after insert or update on rental_contract
for each row
EXECUTE PROCEDURE arch_rental_func();
```

3.Function Creation

```
create function  room_num()
returns trigger as
$BODY$
begin
update hotel set number_of_rooms = ( select count(*) from room where hotel_id = new.hotel_id
) where hotel_id = new.hotel_id;
RETURN new;
end
$BODY$ LANGUAGE plpgsql;


create or replace function  balance_update_new()
returns trigger as
$BODY$
declare days int;
begin
days = new.check_Out_Date - new.check_In_Date;
new.total_cost = (select price from room where hotel_id=new.hotel and room_num = new.room)
* days;
```

```
new.balance = new.total_cost - new.payment_total;
raise notice 'Value: %', new.balance;
RETURN new;
end
$BODY$ LANGUAGE plpgsql;


create or replace function  balance_update()
returns trigger as
$BODY$
declare days int;
begin
days = new.check_Out_Date - new.check_In_Date;
new.total_cost = (select price from room where hotel_id=new.hotel and room_num = new.room)
* days;
new.balance = new.total_cost - new.payment_total;
raise notice 'Value: %', new.balance;
RETURN new;
end
$BODY$ LANGUAGE plpgsql;


create or replace function  payment_update()
returns trigger as
$BODY$
begin
update rental_contract set payment_total = ( select sum(amount_Charged) from payment where
rental_contract = new.rental_contract ) where rental_ID = new.rental_contract;
RETURN new;
end
$BODY$ LANGUAGE plpgsql;
```

```
create function  rollback_entry()
returns trigger as
$BODY$
begin
rollback;
RAISE EXCEPTION 'DATE INCORRECT';
RETURN new;
end
$BODY$ LANGUAGE plpgsql;


create or replace function  arch_book_func()
returns trigger as
$BODY$
Declare arch_id integer;
begin
insert into archive (check_in_date, check_out_date)  values  (new.check_in_date,
new.check_out_date);
arch_id = (SELECT archive_id FROM archive ORDER BY archive_id DESC LIMIT 1);
insert into archive_booking (archive_id, booking_id)  values  (arch_id, new.booking_id);
insert into archive_room(archive_id, hotel,room) values (arch_id, new.hotel, new.room);
insert into archive_customer(archive_id, customer_email) values (arch_id, new.customer);
RETURN new;
end
$BODY$ LANGUAGE plpgsql;
```

```
create or replace function  arch_rental_func()
returns trigger as
$BODY$
Declare arch_id integer;
begin
insert into archive (check_in_date, check_out_date)  values  (new.check_in_date,
new.check_out_date);
arch_id = (SELECT archive_id FROM archive ORDER BY archive_id DESC LIMIT 1);
insert into archive_contract (archive_id, contract_id)  values  (arch_id, new.rental_id);
insert into archive_room(archive_id, hotel,room) values (arch_id, new.hotel, new.room);
insert into archive_customer(archive_id, customer_email) values (arch_id, new.customer);
insert into Archive_checkin(archive_id, check_in_empl) values (arch_id, new.check_in_empl);
RETURN new;
end
$BODY$ LANGUAGE plpgsql;
```

4.View Creation

```
Create view checked_in_rooms as select H.hotel_id, H.chain, H.name, RO.room_num,
RC.customer, RC.check_In_Date,
RC.check_Out_Date from Hotel H, Room RO, Rental_contract RC
where H.hotel_ID = RO.hotel_ID and RO.room_num = RC.room and RO.hotel_ID = RC.hotel;

Create view booked_rooms as select H.hotel_id, H.chain, H.name, RO.room_num, BO.customer,
BO.check_In_Date,
BO.check_Out_Date from Hotel H, Room RO, Booking BO
where H.hotel_ID = RO.hotel_ID and RO.room_num = BO.room and RO.hotel_ID = BO.hotel;

Create view checked_booked_in_rooms as select * from checked_in_rooms
UNION
select * from booked_rooms;
```

```
drop view all_rooms;
Create view all_rooms as select H.chain, H.name Hotel, H.rating, H.number_of_rooms, RO.*,
H.address_prov province,H.address_city city
from Room RO, hotel H where RO.hotel_id = H.hotel_id;

create view room_sizes_available as
Select  DISTINCT H.name, R.capacity from hotel H, room R where H.hotel_id = R.hotel_id;

create view number_of_rooms_per_area as
Select city, count(*) as number_of_rooms from all_rooms group by city;
```

5. Sample queries

```
select *
from all_rooms
where chain = 'Marriot' and (hotel_id, room_num)  not in
(select  C.hotel_id, C.room_num
from checked_booked_in_rooms C where (C.check_Out_Date BETWEEN '2019-04-07' and
'2019-04-17') or (C.check_In_Date BETWEEN '2019-04-07' and '2019-04-17') );
where (C.check_Out_Date >= '2019-04-07' and C.check_In_Date <= '2019-04-07') or
(C.check_Out_Date >= '2019-04-17' and C.check_In_Date <= '2019-04-17');

select * from all_rooms;

--latest booking
SELECT Booking_id FROM Booking ORDER BY Booking_id DESC LIMIT 1;

select distinct(capacity) from all_rooms where hotel='Hilton Ottawa';
```

And others. See file SQL_CODE_to_support_func.sql