CSE 4/560

Databases and Query Languages

Project Milestone 1

Entertainment Booking and Purchase Management System

1. Project Detail

Project Name: Entertainment Booking and Purchase Management System

Name of the Team: Riddham, Chitrak, and Hrushikesh DMQL Project

Team Members	UB ID
Riddham Suvagiya (50466618)	riddhamr
Chitrak Vimalbhai Dave (50478004)	chitrakv
Hrushikesh Choudhary (50482411)	choudha4

2. Problem Statement

2.1 Description of the Problem

The management systems for different types of event can be very challenging. The primary purpose of creating the database for the entertainment booking and purchase managemnet system is to make the entire system of organizing and managing such events easy and in real-time. There are a number of benefits of using this database for such events. This system gets updated automatically so you can keep track of the number of tickets that have been sold out, tickets that are left, the number of presents and future events, and manage the payment methods with great ease. So, you just sit and relax and let this system do all hard work for you.

2.2 The comparison between database systems and excel files

There are several disadvantages of using excel that databases can handle efficiently like accuracy, security, and size.

Size:

- Excel files allow complex calculation between multiple sets of data but it becomes very difficult to manage it.
- While databases can support an unlimited number of tables and allow us to work with them very efficiently using SQL queries.

Accuracy:

- Imagine that there is a column named city in the excel file. We want to change the name of a city. Now when we go for update the value in multiple places there is a probability of making errors. We can do this with ctrl F but it may happen that we may miss one.
- While in case of databases using SQL and 3rd Normal Form we can make such changes very easily.

Security:

- There is a very high security in databases. Excel sheets provide users with the binary level of access either we have file or link or we don't.
- While in databases there are complex permission to prevent people from even being able to see certain parts of data. It also allows us to encrypt the data in the database. There are many SQL queries with which we can grant different privileges to different types of users which ultimately increases the level of security.

3. Target User

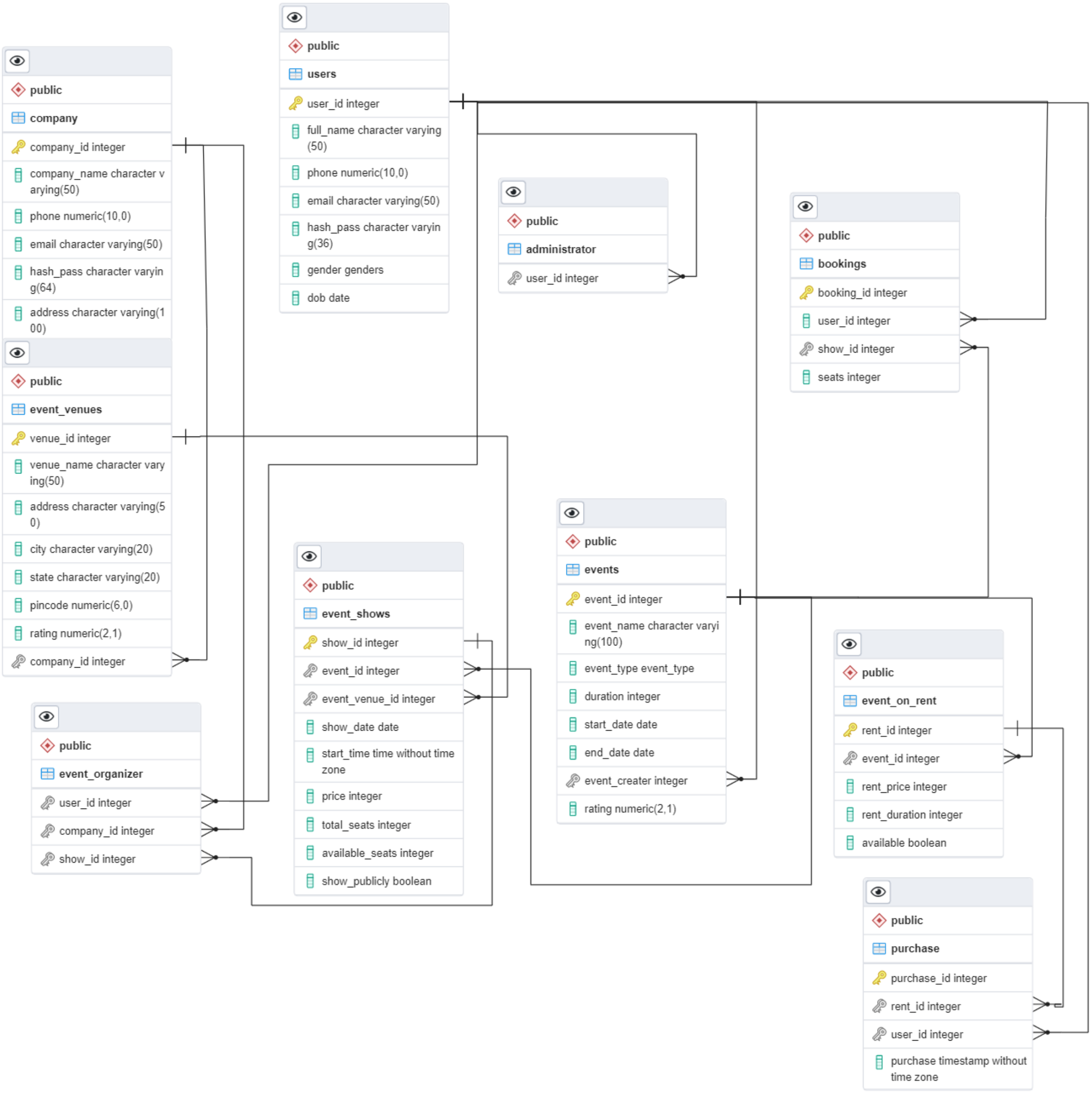
Event managers: They are the primary users of the database. They can update the information of customers and also add/remove them.

Event organizer: They can update the information of events in the database. They organize new events and also can add/remove old events.

Customers: They can only view all the information about events and can purchase tickets if they want.

Database administrator (Admin): They can add/remove managers, events, and users.

4. E/R diagram

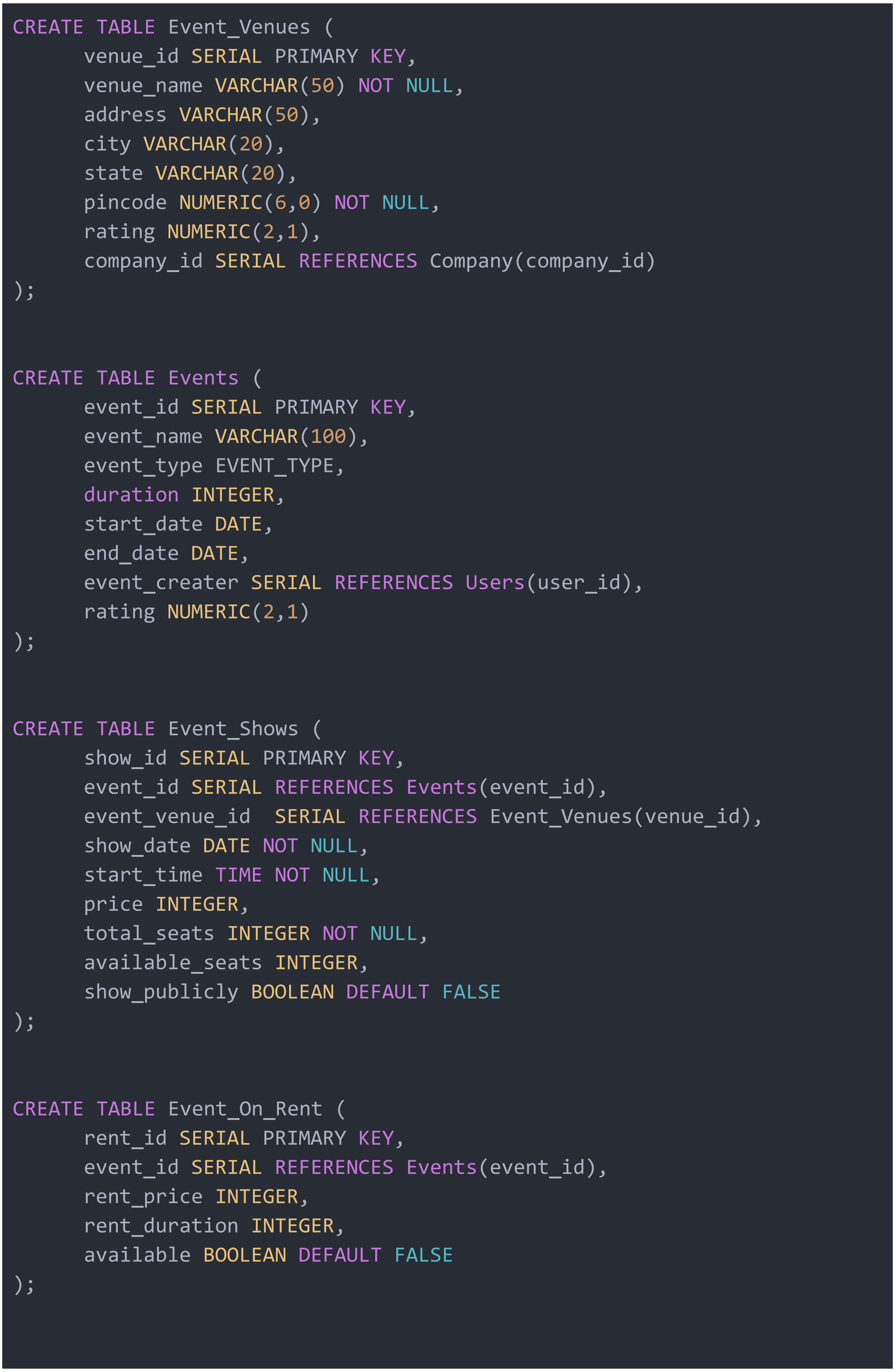

public
users
user_id integer
full_name character varying (50)
phone numeric(10,0)
email character varying(50)
hash_pass character varying(36)
gender genders
dob date
public
company
company_id integer
company_name character varying(50)
phone numeric(10,0)
email character varying(50)
hash_pass character varying(64)
address character varying(100)
public
administrator
user_id integer
public
bookings
booking_id integer
user_id integer
show_id integer
seats integer
public
event_venues
venue_id integer
venue_name character varying(50)
address character varying(50)
city character varying(20)
state character varying(20)
pincode numeric(6,0)
rating numeric(2,1)
company_id integer
public
events
event_id integer
event_name character varying(100)
event_type event_type
duration integer
start_date date
end_date date
event_creater integer
rating numeric(2,1)
public
event_shows
show_id integer
event_id integer
event_venue_id integer
show_date date
start_time time without time zone
price integer
total_seats integer
available_seats integer
show_publicly boolean
public
event_on_rent
rent_id integer
event_id integer
rent_price integer
rent_duration integer
available boolean
public
event_organizer
user_id integer
company_id integer
show_id integer
public
purchase
purchase_id integer
rent_id integer
user_id integer
purchase timestamp without time zone

5. Database Implementation

5.1 Data Schemas

1. Event_Venues (venue_id:integer, vanue_name: varchar(50), address:varchar(50), city: varchar(20), state: varchar(20), pincode: numeric(6,0) , rating : numeric(2,1), company_id: integer)
2. Events – (event_id: integer, event_name: varcahr(30),event_type: enum, duration: integer, start_date: date, end_date: date, event_creator: integer, rating: numeric(2,0))
3. Events_shows – (show_id: integer, event_id: integer, event_vanue_id: integer, show_date: date, start_time: date , price: integer, total_seats: integer, available_seats: integer, show_publicly: boolean)
4. Event_on_Rent – (rent_id: integer, event_id: integer, rent_price: integer, rent_duration: integer, available: boolean)
5. Purchase – (purchase id: integer, rent id: integer, user id: integer, purchase: timestamp)
6. Bookings - (booking_id: integer, user id integer:, event id: integer, seats: integer)
7. Company – (company_id: integer, company_name: varchar(50), phone: numeric(10,0), email: varchar(30), hash_pass: varchar(64), address: varchar(100))
8. Users (user_id: integer, full_name: varchar(50), phone: numeric(10,0), email: varchar (30), hash_pass: varchar(64), gender: enum, dob: date)
9. Administrator - (user_id: integer)
10. Event_organizer– (user_id: integer, company_id: integer, show_id: integer)

```
CREATE TYPE EVENT_TYPE AS ENUM ('Movie', 'Play', 'Concert', 'Stand-up');
CREATE TYPE GENDERS AS ENUM ('Male', 'Female', 'Other', 'Not to
Mention');

CREATE TABLE Company (
      company_id SERIAL PRIMARY KEY,
      company_name VARCHAR(50),
      phone NUMERIC(10, 0),
      email VARCHAR(50) NOT NULL,
      hash_pass VARCHAR(36),
      address VARCHAR(100)
);

CREATE TABLE Users (
      user_id SERIAL PRIMARY KEY,
      full_name VARCHAR(50),
      phone NUMERIC(10, 0),
      email VARCHAR(50) NOT NULL,
      hash_pass VARCHAR(36),
      gender GENDERS,
      dob DATE NOT NULL
);
```



```
CREATE TABLE Event_Venues (
      venue_id SERIAL PRIMARY KEY,
      venue_name VARCHAR(50) NOT NULL,
      address VARCHAR(50),
      city VARCHAR(20),
      state VARCHAR(20),
      pincode NUMERIC(6,0) NOT NULL,
      rating NUMERIC(2,1),
      company_id SERIAL REFERENCES Company(company_id)
);


CREATE TABLE Events (
      event_id SERIAL PRIMARY KEY,
      event_name VARCHAR(100),
      event_type EVENT_TYPE,
      duration INTEGER,
      start_date DATE,
      end_date DATE,
      event_creater SERIAL REFERENCES Users(user_id),
      rating NUMERIC(2,1)
);


CREATE TABLE Event_Shows (
      show_id SERIAL PRIMARY KEY,
      event_id SERIAL REFERENCES Events(event_id),
      event_venue_id  SERIAL REFERENCES Event_Venues(venue_id),
      show_date DATE NOT NULL,
      start_time TIME NOT NULL,
      price INTEGER,
      total_seats INTEGER NOT NULL,
      available_seats INTEGER,
      show_publicly BOOLEAN DEFAULT FALSE
);


CREATE TABLE Event_On_Rent (
      rent_id SERIAL PRIMARY KEY,
      event_id SERIAL REFERENCES Events(event_id),
      rent_price INTEGER,
      rent_duration INTEGER,
      available BOOLEAN DEFAULT FALSE
);
```

```
CREATE TABLE Purchase (
      purchase_id SERIAL PRIMARY KEY,
      rent_id SERIAL REFERENCES Event_On_Rent(rent_id),
      user_id SERIAL REFERENCES Users(user_id),
      purchase TIMESTAMP
);

CREATE TABLE Bookings (
      booking_id SERIAL PRIMARY KEY,
      user_id SERIAL REFERENCES Users(user_id),
      show_id SERIAL REFERENCES Event_shows(show_id),
      seats INTEGER
);

CREATE TABLE Event_organizer (
      user_id SERIAL REFERENCES Users(user_id),
      company_id SERIAL REFERENCES Company(company_id),
      show_id SERIAL REFERENCES Event_Shows(show_id)
);

CREATE TABLE Administrator (
      user_id SERIAL REFERENCES Users(user_id)
);
```

Relationships between tables:

In relation company company_id is primary key which will act as foreign key in relations event_venues and event_organizer.
In relation event_venues venue_id is primary key which acts as a foreign key in relation event_. In relation Users the primary key is user_id which is foreign key for relation purchase and bookings.
In relation event_shows the primary key is show_id which is foreign key for table event_organizer.
In relation events event_id is primary key and acts as a foreign key in table event_on_rent.
In relation event_on_rent, rent_id is primary key and acts as a foreign key in table purchase.

5.2 Attributes

- Event_Venues: In this table, venue_id is primary key. It specifies unique venue id where events will be held. Venue_name denotes name of venue. Address attribute is for address of venue, state is for state of venue and can be null, pincode represents zip code, rating is for rating given by customers and company_id is foreign key and for the company's id which is organzing the event.

- Events: In Events, event_id is primary key for the id of the event. Event_name is for name of event, duration is for duration of the event, start_type and end_type are for start and end date respectively for events. Event_creator is foreign key and for the creator of the event and rating is numeric type for the rating or feedback by customers.
- Event_shows: in this table, show_id is primary key and for the id of the show. Event_id is foreign key and represents id of the event, event_venue_id is foreign key and for id of the event's venue. Show_data is for the data on which show will be held, start_time is for starting time of the show, price is for the price of the ticket, total_seats is for total seats of show, available_seats represents available seats for customers to book, show_publicly is for whether a show is available for customers to book or not.
- event _on_rent: In event_on_rent , rent_id is primary key and it is for different types of rent periods. If an events is on rent for 1 hour, 1 week and 1 month then all these cases will have different rent_id. Event_id is foreign key and for id of the event, rent_price is for price for rent for different periods of time. Rent_duration is for time duration for which you want to rent an event. Available is to know whether an event is there or not.
- Purchase: In purchase, purchase_id is primary key and after purchasing ticket the user will get this id , rent_id is foreign key and for the , user_id for users who purchases the event.
- Bookings: booking_id is primary key and it is for identifying each booking made by the customer, user_id is for the id of the user who books an event, event_id is foreign key and for id of events, and seats is for number of seats that the user wants to book.
- Company: company_id is primary key and represents the id of the company which organizes the events. Company_name is for name of the company, phone is of numeric type and for phone number of the company, email is for email id of the company, hash_pass is for encoded password to log in the company so nobody can see it. Address is for address of the company and can be null.
- Users: user_id is primary key and for identifying each user uniquely. Full_name is for full name of the user, phone is for phone number of the user and can be null, email is for email id of the user, hash_pass is for password of the users which is in encoded form so noone can see it. Gender is for gender of the user, and dob is for date of birth of the user.
- Administrator: user_id is for the administrator's id.
- Event_organizer: in this table, user_id is for id for the event organizer, company_id is foreign key and for id of the company to which that event organizer belongs and show_id is foreign key and for the id of the show which that organizer organizes.

5.3 Primary and Foreign Keys

- Here most table of the database has a primary key and one or more foreign keys.

```
ALTER TABLE bookings
```

```
ADD CONSTRAINT User_id_fk FOREIGN KEY(user_id)
    REFERENCES Users(user_id)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION;
```

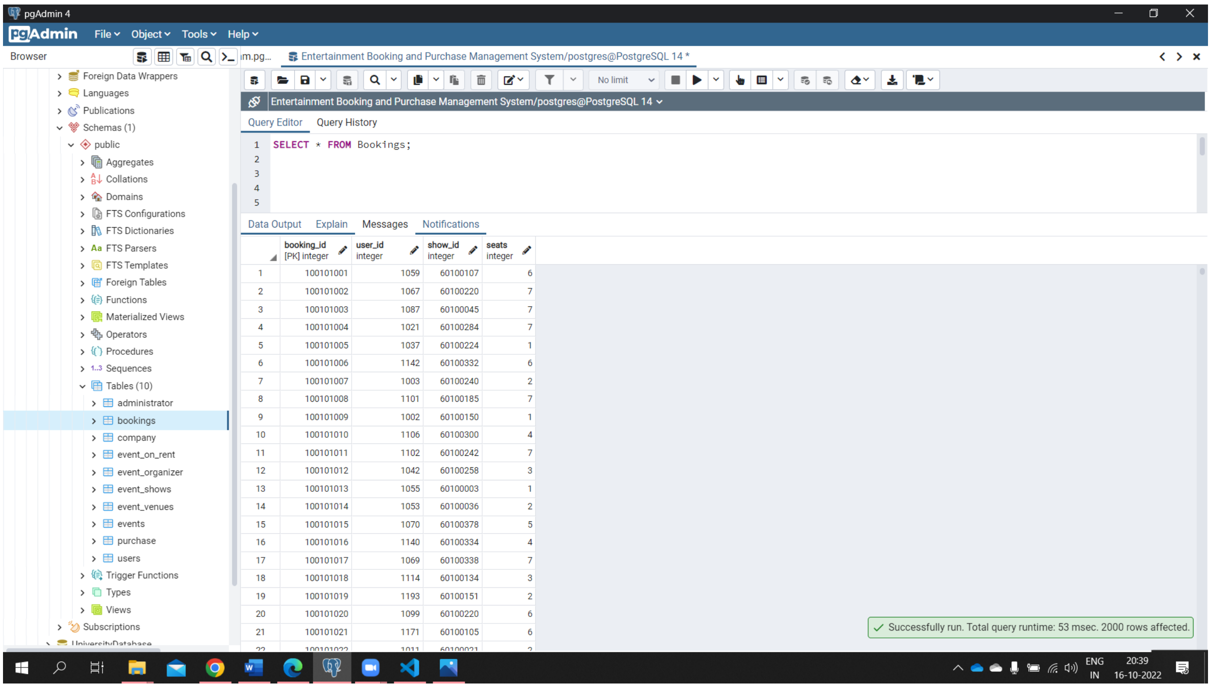
As on delete user we don’t want to remove any bookings that’s why we are setting that as No Action ON DELETE.

```
ALTER TABLE Administrator
ADD CONSTRAINT User_id_fk FOREIGN KEY(user_id)
    REFERENCES Users(user_id)
    ON DELETE CASCADE;
```

As on Delete User we want to remove id from Admin table so we set as cascade

Right now we don’t add constraint such as SET DEFAULT / SET NULL ON DELETE, which means we are planning to either keep the data or remove the data when there is any deletion happen.

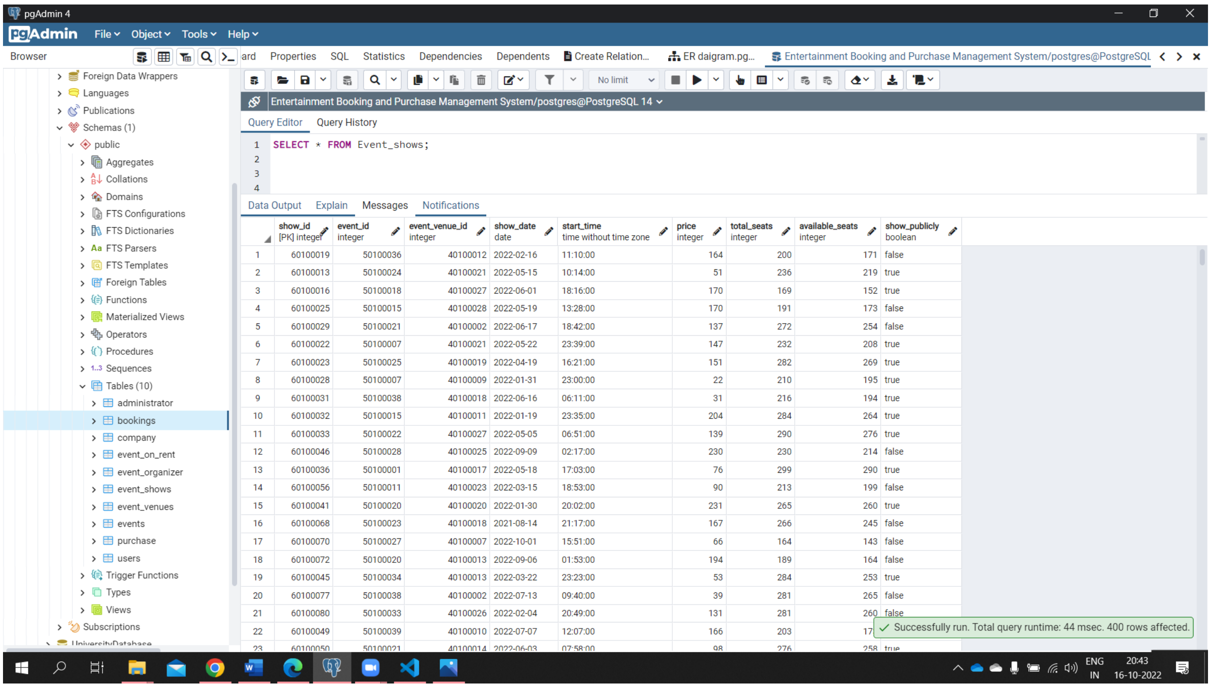
5.4 Records Insertion



```sql
SELECT * FROM Bookings;
```

	booking_id [PK] integer	user_id integer	show_id integer	seats integer
1	100101001	1059	60100107	6
2	100101002	1067	60100220	7
3	100101003	1087	60100045	7
4	100101004	1021	60100284	7
5	100101005	1037	60100224	1
6	100101006	1142	60100332	6
7	100101007	1003	60100240	2
8	100101008	1101	60100185	7
9	100101009	1002	60100150	1
10	100101010	1106	60100300	4
11	100101011	1102	60100242	7
12	100101012	1042	60100258	3
13	100101013	1055	60100003	1
14	100101014	1053	60100036	2
15	100101015	1070	60100378	5
16	100101016	1140	60100334	4
17	100101017	1069	60100338	7
18	100101018	1114	60100134	3
19	100101019	1193	60100151	2
20	100101020	1099	60100220	6
21	100101021	1171	60100105	6
22	100101022	1011	60100021	2

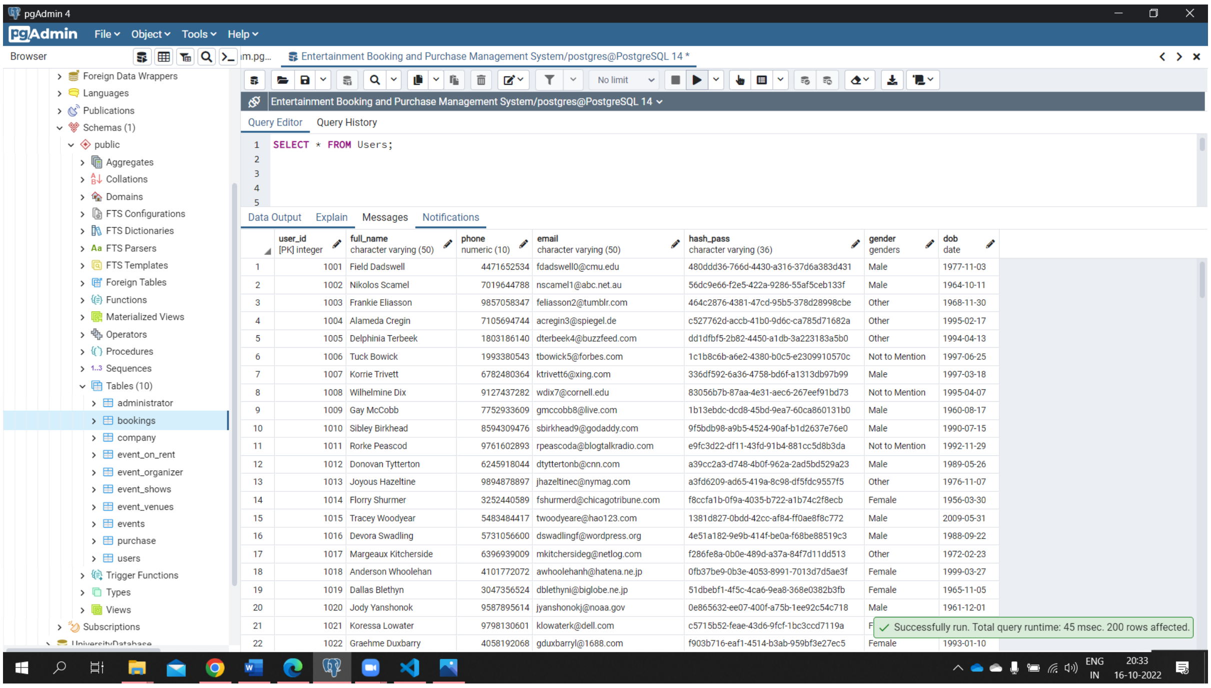
✓ Successfully run. Total query runtime: 53 msec. 2000 rows affected.



```sql
SELECT * FROM Event_shows;
```

	show_id [PK] integer	event_id integer	event_venue_id integer	show_date date	start_time time without time zone	price integer	total_seats integer	available_seats integer	show_publicly boolean
1	60100019	50100036	40100012	2022-02-16	11:10:00	164	200	171	false
2	60100013	50100024	40100021	2022-05-15	10:14:00	51	236	219	true
3	60100016	50100018	40100027	2022-06-01	18:16:00	170	169	152	true
4	60100025	50100015	40100028	2022-05-19	13:28:00	170	191	173	false
5	60100029	50100021	40100002	2022-06-17	18:42:00	137	272	254	false
6	60100022	50100007	40100021	2022-05-22	23:39:00	147	232	208	true
7	60100023	50100025	40100019	2022-04-19	16:21:00	151	282	269	true
8	60100028	50100007	40100009	2022-01-31	23:00:00	22	210	195	true
9	60100031	50100038	40100018	2022-06-16	06:11:00	31	216	194	true
10	60100032	50100015	40100011	2022-01-19	23:35:00	204	284	264	true
11	60100033	50100022	40100027	2022-05-05	06:51:00	139	290	276	true
12	60100046	50100028	40100025	2022-09-09	02:17:00	230	230	214	false
13	60100036	50100001	40100017	2022-05-18	17:03:00	76	299	290	true
14	60100056	50100011	40100023	2022-03-15	18:53:00	90	213	199	false
15	60100041	50100020	40100020	2022-01-30	20:02:00	231	265	260	true
16	60100068	50100023	40100018	2021-08-14	21:17:00	167	266	245	false
17	60100070	50100027	40100007	2022-10-01	15:51:00	66	164	143	false
18	60100072	50100020	40100013	2022-09-06	01:53:00	194	189	164	false
19	60100045	50100034	40100013	2022-03-22	23:23:00	53	284	253	true
20	60100077	50100038	40100002	2022-07-13	09:40:00	39	281	265	false
21	60100080	50100033	40100026	2022-02-04	20:49:00	131	281	260	false
22	60100049	50100039	40100010	2022-07-07	12:07:00	166	203	17	
23	60100050	50100021	40100014	2022-06-03	07:58:00	98	276	258	true

✓ Successfully run. Total query runtime: 44 msec. 400 rows affected.

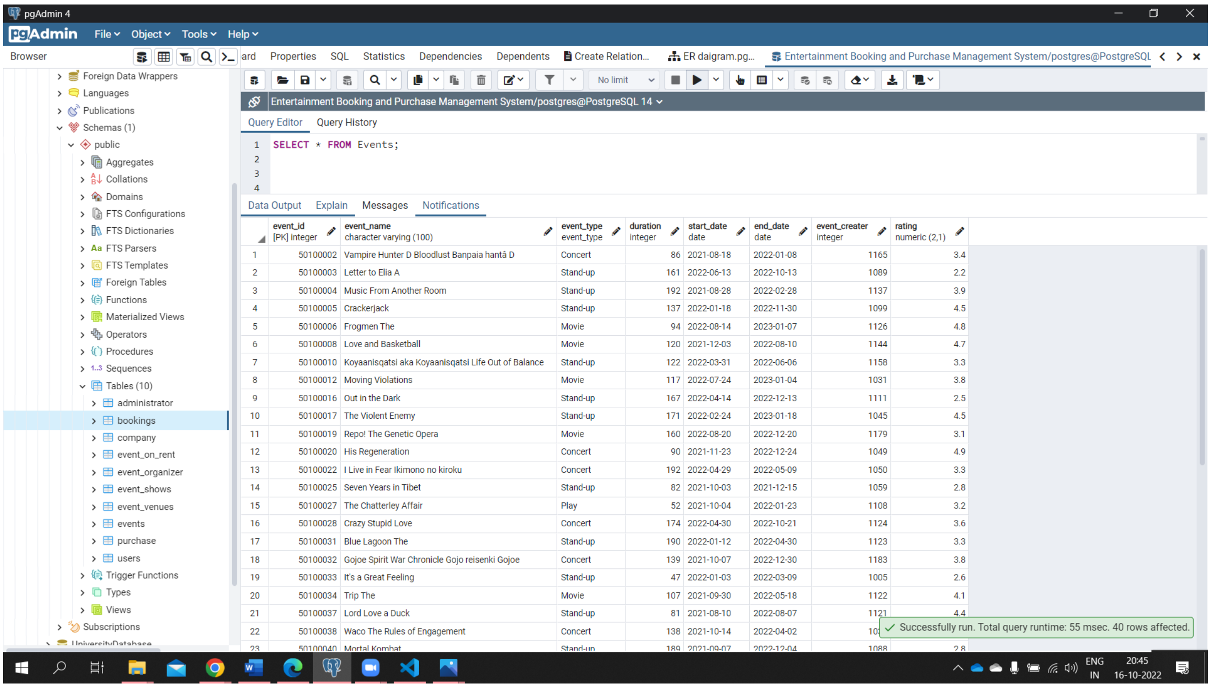


Entertainment Booking and Purchase Management System/postgres@PostgreSQL 14

```
SELECT * FROM Users;
```

	user_id [PK] integer	full_name character varying (50)	phone numeric (10)	email character varying (50)	hash_pass character varying (36)	gender genders	dob date
1	1001	Field Dadswell	4471652534	fdadswell0@cmu.edu	480ddd36-766d-4430-a316-37d6a383d431	Male	1977-11-03
2	1002	Nikolos Scamel	7019644788	nscamel1@abc.net.au	56dc9e66-f2e5-422a-9286-55af5ceb133f	Male	1964-10-11
3	1003	Frankie Eliasson	9857058347	feliasson2@tumblr.com	464c2876-4381-47cd-95b5-378d28998cbe	Other	1968-11-30
4	1004	Alameda Cregin	7105694744	acregin3@spiegel.de	c527762d-accb-41b0-9d6c-ca785d71682a	Other	1995-02-17
5	1005	Delphinia Terbeek	1803186140	dterbeek4@buzzfeed.com	dd1dfbf5-2b82-4450-a1db-3a223183a5b0	Other	1994-04-13
6	1006	Tuck Bowick	1993380543	tbowick5@forbes.com	1c1b8c6b-a6e2-4380-b0c5-e2309910570c	Not to Mention	1997-06-25
7	1007	Korrie Trivett	6782480364	ktrivett6@xing.com	336df592-6a36-4758-bd6f-a1313db97b99	Male	1997-03-18
8	1008	Wilhelmine Dix	9127437282	wdix7@cornell.edu	83056b7b-87aa-4e31-aec6-267eef91bd73	Not to Mention	1995-04-07
9	1009	Gay McCobb	7752933609	gmccobb8@live.com	1b13ebdc-dcd8-45bd-9ea7-60ca860131b0	Male	1960-08-17
10	1010	Sibley Birkhead	8594309476	sbirkhead9@godaddy.com	9f5bdb98-a9b5-4524-90af-b1d2637e76e0	Male	1990-07-15
11	1011	Rorke Peascod	9761602893	rpeascoda@blogtalkradio.com	e9fc3d22-df11-43fd-91b4-881cc5d8b3da	Not to Mention	1992-11-29
12	1012	Donovan Tytterton	6245918044	dtyttertonb@cnn.com	a39cc2a3-d748-4b0f-962a-2ad5bd529a23	Male	1989-05-26
13	1013	Joyous Hazeltine	9894878897	jhazeltinec@nymag.com	a3fd6209-ad65-419a-8c98-df5fdc9557f5	Other	1976-11-07
14	1014	Florry Shurmer	3252440589	fshurmerd@chicagotribune.com	f8ccfa1b-0f9a-4035-b722-a1b74c2f8ecb	Female	1956-03-30
15	1015	Tracey Woodyear	5483484417	twoodyeare@hao123.com	1381d827-0bdd-42cc-af84-ff0ae8f8c772	Male	2009-05-31
16	1016	Devora Swadling	5731056600	dswadlingf@wordpress.org	4e51a182-9e9b-414f-be0a-f68be88519c3	Male	1988-09-22
17	1017	Margeaux Kitcherside	6396939009	mkitchersideg@netlog.com	f286fe8a-0b0e-489d-a37a-84f7d11dd513	Other	1972-02-23
18	1018	Anderson Whoolehan	4101772072	awhoolehanh@hatena.ne.jp	0fb37be9-0b3e-4053-8991-7013d7d5ae3f	Female	1999-03-27
19	1019	Dallas Blethyn	3047356524	dblethyni@biglobe.ne.jp	51dbebf1-4f5c-4ca6-9ea8-368e0382b3fb	Female	1965-11-05
20	1020	Jody Yanshonok	9587895614	jyanshonokj@noaa.gov	0e865632-ee07-400f-a75b-1ee92c54c718	Male	1961-12-01
21	1021	Koressa Lowater	9798130601	klowaterk@dell.com	c5715b52-feae-43d6-9fcf-1bc3ccd7119a	F	
22	1022	Graehme Duxbarry	4058192068	gduxbarryl@1688.com	f903b716-eaf1-4514-b3ab-959bf3e27ec5	Female	1993-01-10

✓ Successfully run. Total query runtime: 45 msec. 200 rows affected.

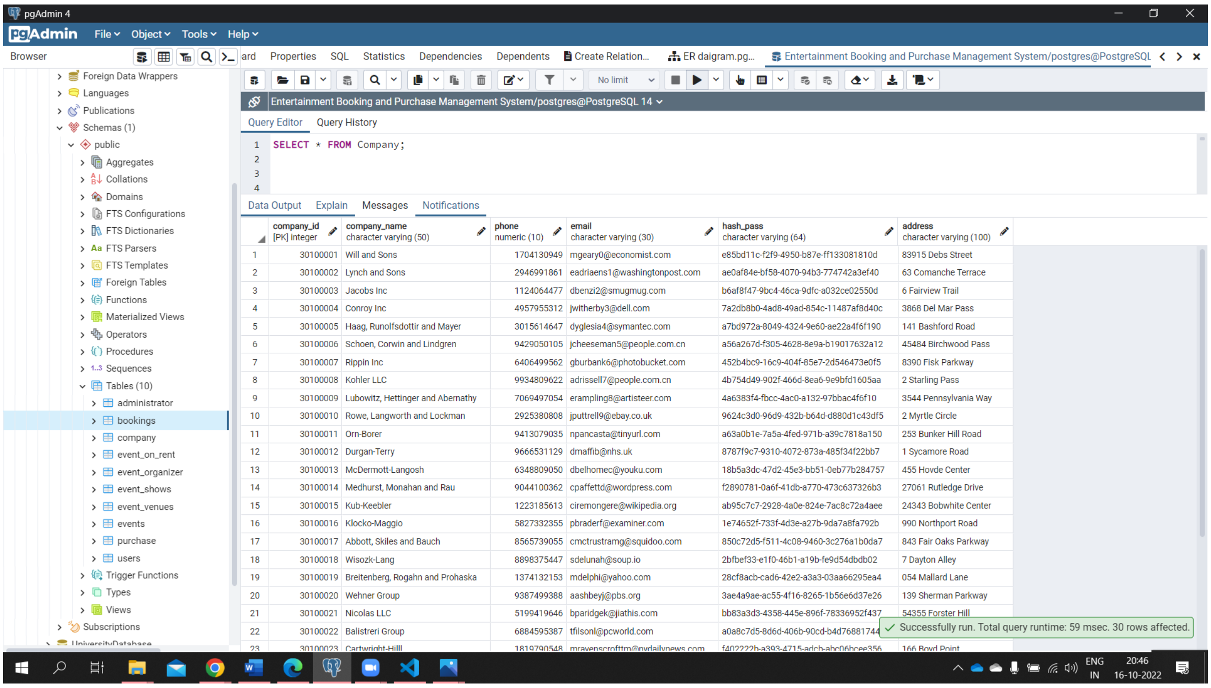


Entertainment Booking and Purchase Management System/postgres@PostgreSQL 14

```
SELECT * FROM Events;
```

	event_id [PK] integer	event_name character varying (100)	event_type event_type	duration integer	start_date date	end_date date	event_creater integer	rating numeric (2,1)
1	50100002	Vampire Hunter D Bloodlust Banpaia hantâ D	Concert	86	2021-08-18	2022-01-08	1165	3.4
2	50100003	Letter to Elia A	Stand-up	161	2022-06-13	2022-10-13	1089	2.2
3	50100004	Music From Another Room	Stand-up	192	2021-08-28	2022-02-28	1137	3.9
4	50100005	Crackerjack	Stand-up	137	2022-01-18	2022-11-30	1099	4.5
5	50100006	Frogmen The	Movie	94	2022-08-14	2023-01-07	1126	4.8
6	50100008	Love and Basketball	Movie	120	2021-12-03	2022-08-10	1144	4.7
7	50100010	Koyaanisqatsi aka Koyaanisqatsi Life Out of Balance	Stand-up	122	2022-03-31	2022-06-06	1158	3.3
8	50100012	Moving Violations	Movie	117	2022-07-24	2023-01-04	1031	3.8
9	50100016	Out in the Dark	Stand-up	167	2022-04-14	2022-12-13	1111	2.5
10	50100017	The Violent Enemy	Stand-up	171	2022-02-24	2023-01-18	1045	4.5
11	50100019	Repo! The Genetic Opera	Movie	160	2022-08-20	2022-12-20	1179	3.1
12	50100020	His Regeneration	Concert	90	2021-11-23	2022-12-24	1049	4.9
13	50100022	I Live in Fear Ikimono no kiroku	Concert	192	2022-04-29	2022-05-09	1050	3.3
14	50100025	Seven Years in Tibet	Stand-up	82	2021-10-03	2021-12-15	1059	2.8
15	50100027	The Chatterley Affair	Play	52	2021-10-04	2022-01-23	1108	3.2
16	50100028	Crazy Stupid Love	Concert	174	2022-04-30	2022-10-21	1124	3.6
17	50100031	Blue Lagoon The	Stand-up	190	2022-01-12	2022-04-30	1123	3.3
18	50100032	Gojoe Spirit War Chronicle Gojo reisenki Gojoe	Concert	139	2021-10-07	2022-12-30	1183	3.8
19	50100033	It's a Great Feeling	Stand-up	47	2022-01-03	2022-03-09	1005	2.6
20	50100034	Trip The	Movie	107	2021-09-30	2022-05-18	1122	4.1
21	50100037	Lord Love a Duck	Stand-up	81	2021-08-10	2022-08-07	1121	4.4
22	50100038	Waco The Rules of Engagement	Concert	138	2021-10-14	2022-04-02	10	
23	50100040	Mortal Kombat	Stand-up	189	2021-09-07	2022-12-04	1088	2.8

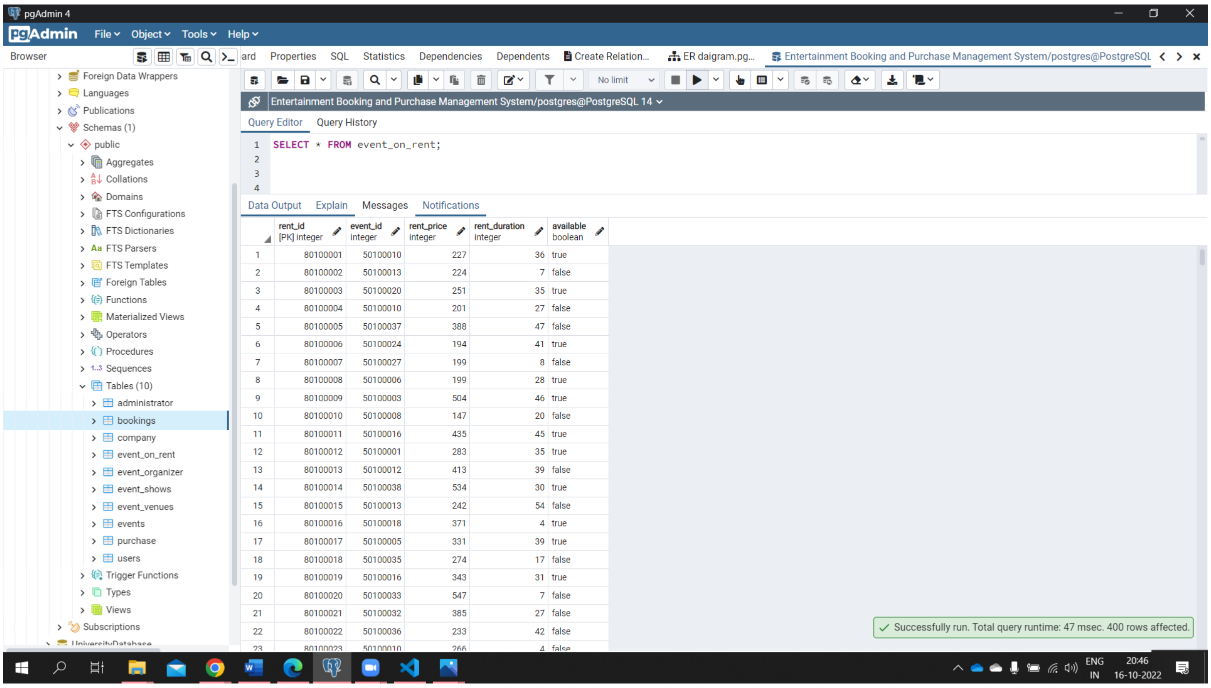
✓ Successfully run. Total query runtime: 55 msec. 40 rows affected.



```
SELECT * FROM Company;
```

	company_id [PK] integer	company_name character varying (50)	phone numeric (10)	email character varying (30)	hash_pass character varying (64)	address character varying (100)
1	30100001	Will and Sons	1704130949	mgeary0@economist.com	e85bd11c-f2f9-4950-b87e-ff133081810d	83915 Debs Street
2	30100002	Lynch and Sons	2946991861	eadriaens1@washingtonpost.com	ae0af84e-bf58-4070-94b3-774742a3ef40	63 Comanche Terrace
3	30100003	Jacobs Inc	1124064477	dbenzi2@smugmug.com	b6af8f47-9bc4-46ca-9dfc-a032ce02550d	6 Fairview Trail
4	30100004	Conroy Inc	4957955312	jwitherby3@dell.com	7a2db8b0-4ad8-49ad-854c-11487af8d40c	3868 Del Mar Pass
5	30100005	Haag, Runolfsdottir and Mayer	3015614647	dyglesia4@symantec.com	a7bd972a-8049-4324-9e60-ae22a4f6f190	141 Bashford Road
6	30100006	Schoen, Corwin and Lindgren	9429050105	jcheeseman5@people.com.cn	a56a267d-f305-4628-8e9a-b19017632a12	45484 Birchwood Pass
7	30100007	Rippin Inc	6406499562	gburbank6@photobucket.com	452b4bc9-16c9-404f-85e7-2d546473e0f5	8390 Fisk Parkway
8	30100008	Kohler LLC	9934809622	adrissell7@people.com.cn	4b754d49-902f-466d-8ea6-9e9bfd1605aa	2 Starling Pass
9	30100009	Lubowitz, Hettinger and Abernathy	7069497054	erampling8@artisteer.com	4a6383f4-fbcc-4ac0-a132-97bbac4f6f10	3544 Pennsylvania Way
10	30100010	Rowe, Langworth and Lockman	2925380808	jputtrell9@ebay.co.uk	9624c3d0-96d9-432b-b64d-d880d1c43df5	2 Myrtle Circle
11	30100011	Orn-Borer	9413079035	npancasta@tinyurl.com	a63a0b1e-7a5a-4fed-971b-a39c7818a150	253 Bunker Hill Road
12	30100012	Durgan-Terry	9666531129	dmaffib@nhs.uk	8787f9c7-9310-4072-873a-485f34f22bb7	1 Sycamore Road
13	30100013	McDermott-Langosh	6348809050	dbelhomec@youku.com	18b5a3dc-47d2-45e3-bb51-0eb77b284757	455 Hovde Center
14	30100014	Medhurst, Monahan and Rau	9044100362	cpaffettd@wordpress.com	f2890781-0a6f-41db-a770-473c637326b3	27061 Rutledge Drive
15	30100015	Kub-Keebler	1223185613	ciremongere@wikipedia.org	ab95c7c7-2928-4a0e-824e-7ac8c72a4aee	24343 Bobwhite Center
16	30100016	Klocko-Maggio	5827332355	pbraderf@examiner.com	1e74652f-733f-4d3e-a27b-9da7a8fa792b	990 Northport Road
17	30100017	Abbott, Skiles and Bauch	8565739055	cmctrustramg@squidoo.com	850c72d5-f511-4c08-9460-3c276a1b0da7	843 Fair Oaks Parkway
18	30100018	Wisozk-Lang	8898375447	sdelunah@soup.io	2bfbef33-e1f0-46b1-a19b-fe9d54dbdb02	7 Dayton Alley
19	30100019	Breitenberg, Rogahn and Prohaska	1374132153	mdelphi@yahoo.com	28cf8acb-cad6-42e2-a3a3-03aa66295ea4	054 Mallard Lane
20	30100020	Wehner Group	9387499388	aashbeyj@pbs.org	3ae4a9ae-ac55-4f16-8265-1b56e6d37e26	139 Sherman Parkway
21	30100021	Nicolas LLC	5199419646	bparidgek@jiathis.com	bb83a3d3-4358-445e-896f-78336952f437	54355 Forster Hill
22	30100022	Balistreri Group	6884595387	tfilsonl@pcworld.com	a0a8c7d5-8d6d-406b-90cd-b4d76881744	
23	30100023	Cartwright-Hilll	1819790548	mravenscrofttm@nydailynews.com	f402222b-a393-4715-adcb-abc06bcee356	166 Boyd Point

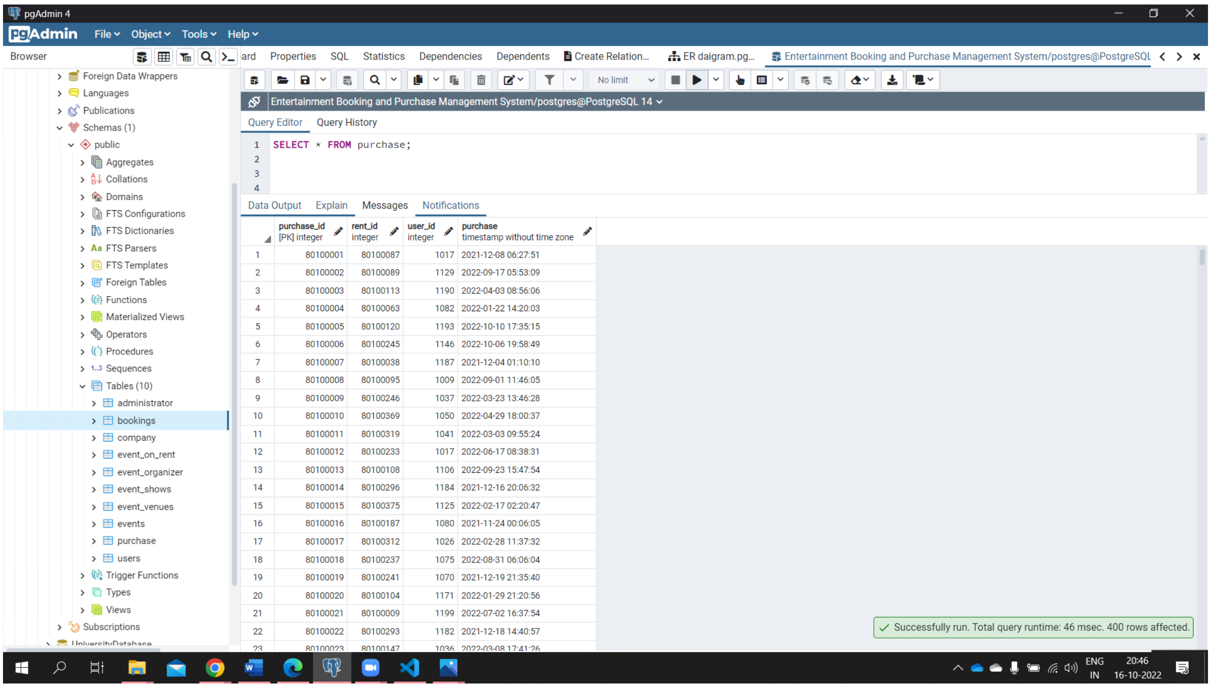
✓ Successfully run. Total query runtime: 59 msec. 30 rows affected.



```
SELECT * FROM event_on_rent;
```

	rent_id [PK] integer	event_id integer	rent_price integer	rent_duration integer	available boolean
1	80100001	50100010	227	36	true
2	80100002	50100013	224	7	false
3	80100003	50100020	251	35	true
4	80100004	50100010	201	27	false
5	80100005	50100037	388	47	false
6	80100006	50100024	194	41	true
7	80100007	50100027	199	8	false
8	80100008	50100006	199	28	true
9	80100009	50100003	504	46	true
10	80100010	50100008	147	20	false
11	80100011	50100016	435	45	true
12	80100012	50100001	283	35	true
13	80100013	50100012	413	39	false
14	80100014	50100038	534	30	true
15	80100015	50100013	242	54	false
16	80100016	50100018	371	4	true
17	80100017	50100005	331	39	true
18	80100018	50100035	274	17	false
19	80100019	50100016	343	31	true
20	80100020	50100033	547	7	false
21	80100021	50100032	385	27	false
22	80100022	50100036	233	42	false
23	80100023	50100010	266	4	false

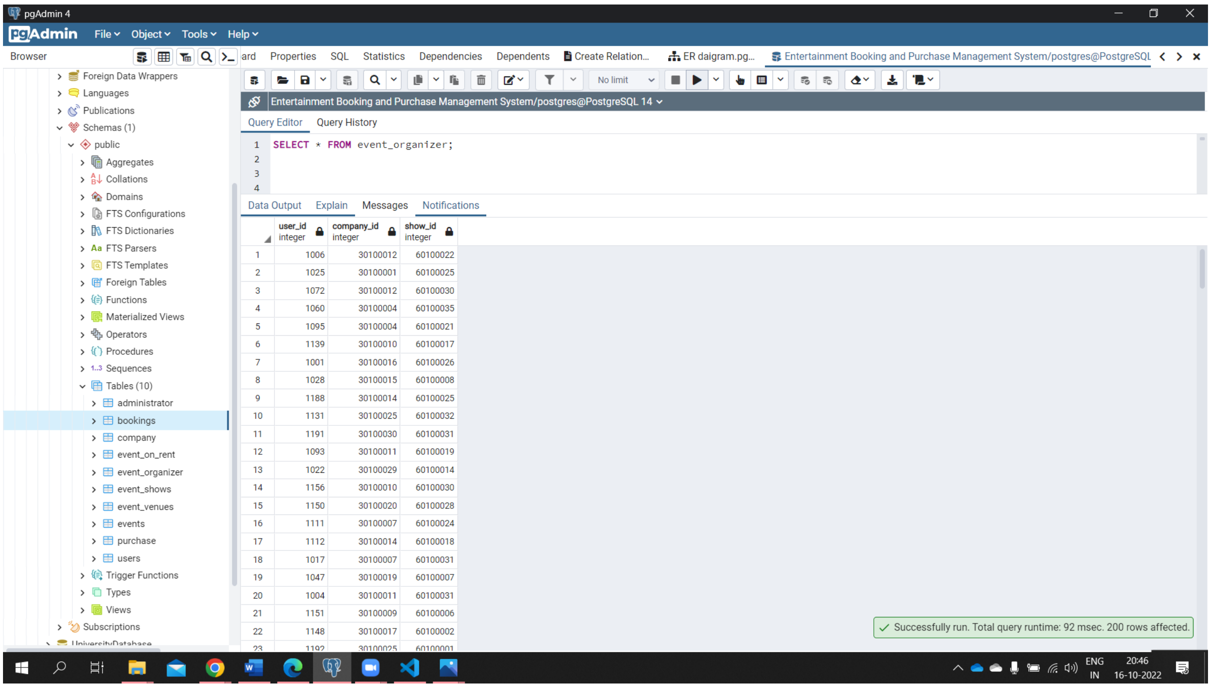
✓ Successfully run. Total query runtime: 47 msec. 400 rows affected.



```
SELECT * FROM purchase;
```

	purchase_id [PK] integer	rent_id integer	user_id integer	purchase timestamp without time zone
1	80100001	80100087	1017	2021-12-08 06:27:51
2	80100002	80100089	1129	2022-09-17 05:53:09
3	80100003	80100113	1190	2022-04-03 08:56:06
4	80100004	80100063	1082	2022-01-22 14:20:03
5	80100005	80100120	1193	2022-10-10 17:35:15
6	80100006	80100245	1146	2022-10-06 19:58:49
7	80100007	80100038	1187	2021-12-04 01:10:10
8	80100008	80100095	1009	2022-09-01 11:46:05
9	80100009	80100246	1037	2022-03-23 13:46:28
10	80100010	80100369	1050	2022-04-29 18:00:37
11	80100011	80100319	1041	2022-03-03 09:55:24
12	80100012	80100233	1017	2022-06-17 08:38:31
13	80100013	80100108	1106	2022-09-23 15:47:54
14	80100014	80100296	1184	2021-12-16 20:06:32
15	80100015	80100375	1125	2022-02-17 02:20:47
16	80100016	80100187	1080	2021-11-24 00:06:05
17	80100017	80100312	1026	2022-02-28 11:37:32
18	80100018	80100237	1075	2022-08-31 06:06:04
19	80100019	80100241	1070	2021-12-19 21:35:40
20	80100020	80100104	1171	2022-01-29 21:20:56
21	80100021	80100009	1199	2022-07-02 16:37:54
22	80100022	80100293	1182	2021-12-18 14:40:57
23	80100023	80100147	1036	2022-03-08 17:41:26

✓ Successfully run. Total query runtime: 46 msec. 400 rows affected.

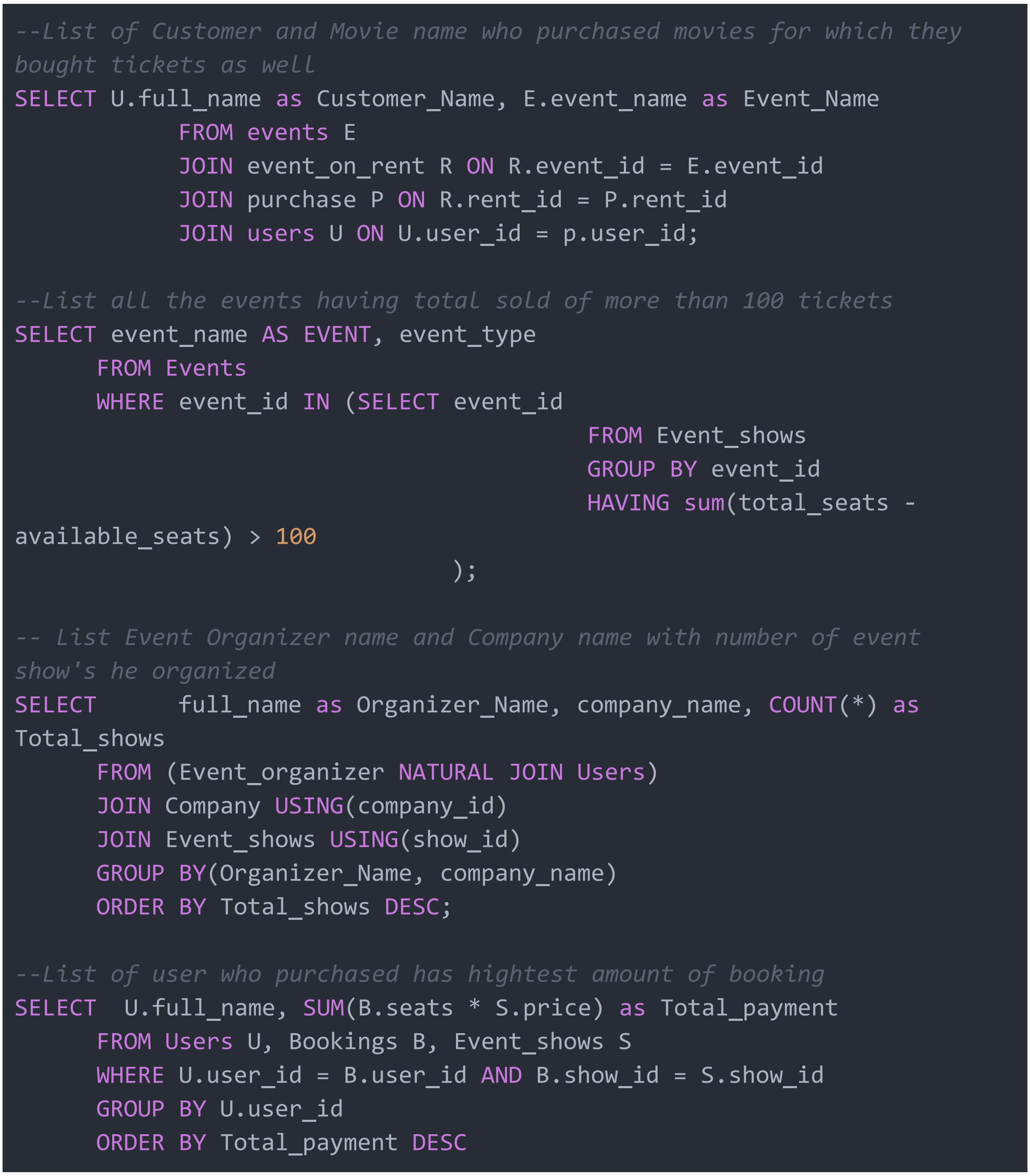


```
SELECT * FROM event_organizer;
```

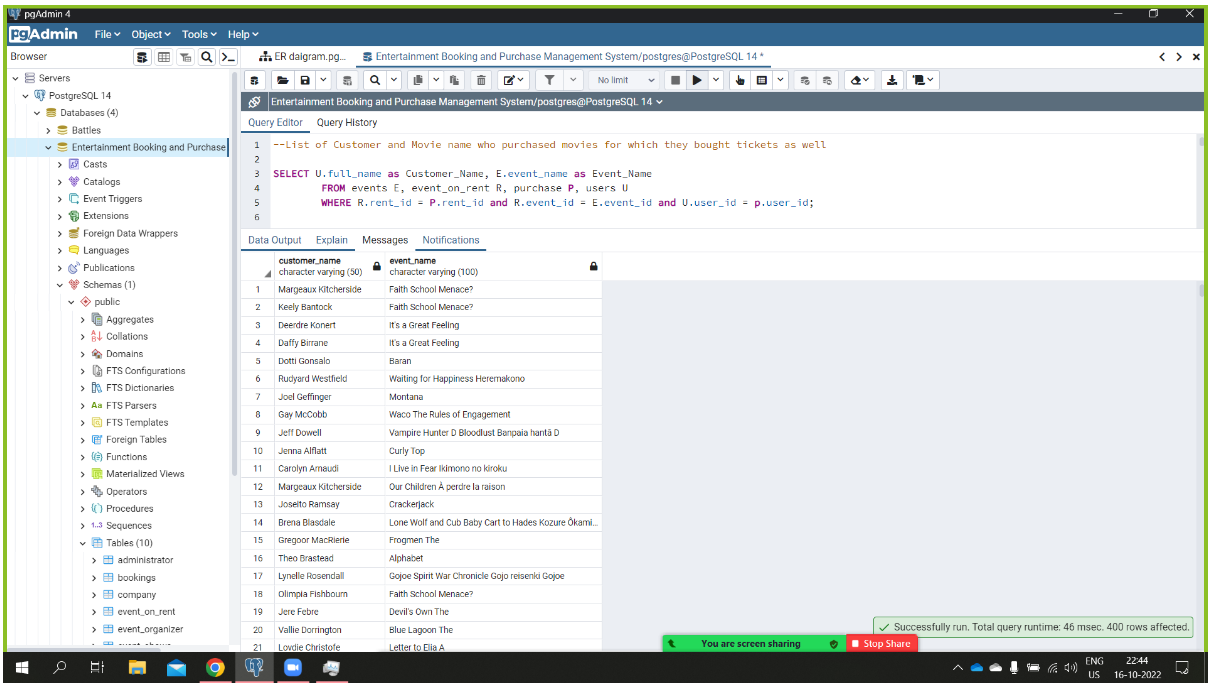
	user_id integer	company_id integer	show_id integer
1	1006	30100012	60100022
2	1025	30100001	60100025
3	1072	30100012	60100030
4	1060	30100004	60100035
5	1095	30100004	60100021
6	1139	30100010	60100017
7	1001	30100016	60100026
8	1028	30100015	60100008
9	1188	30100014	60100025
10	1131	30100025	60100032
11	1191	30100030	60100031
12	1093	30100011	60100019
13	1022	30100029	60100014
14	1156	30100010	60100030
15	1150	30100020	60100028
16	1111	30100007	60100024
17	1112	30100014	60100018
18	1017	30100007	60100031
19	1047	30100019	60100007
20	1004	30100011	60100031
21	1151	30100009	60100006
22	1148	30100017	60100002
23	1192	30100025	60100001

✓ Successfully run. Total query runtime: 92 msec. 200 rows affected.

6. Queries



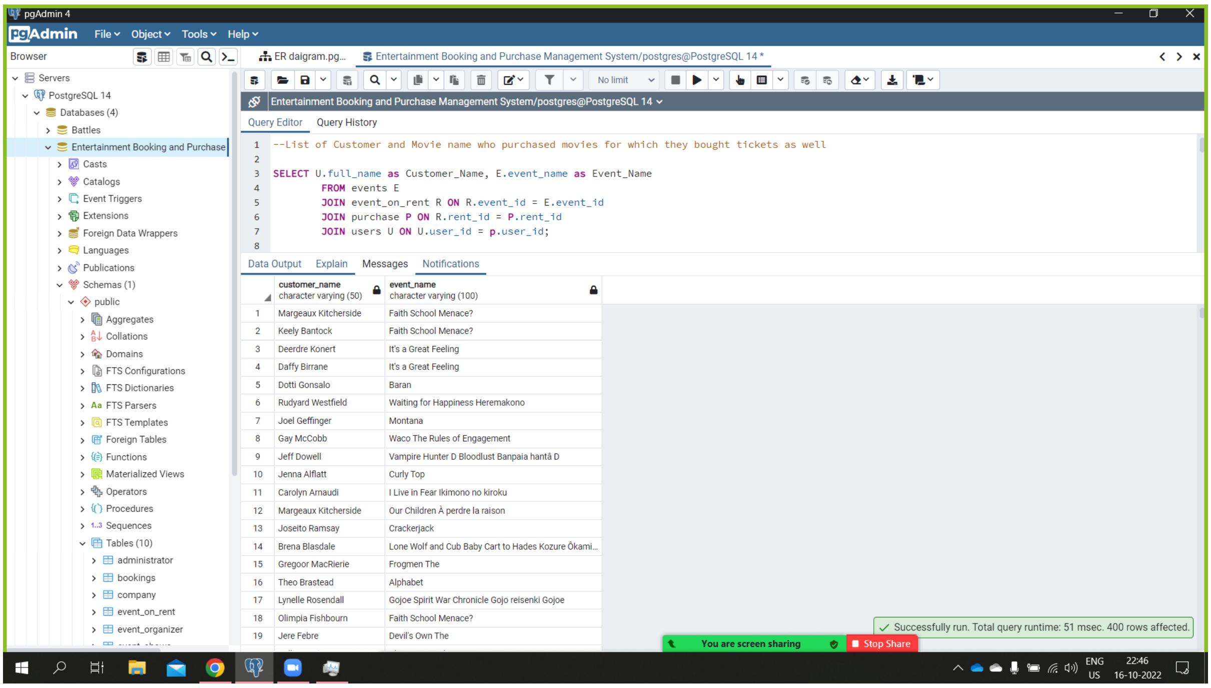
```sql
--List of Customer and Movie name who purchased movies for which they
bought tickets as well
SELECT U.full_name as Customer_Name, E.event_name as Event_Name
            FROM events E
            JOIN event_on_rent R ON R.event_id = E.event_id
            JOIN purchase P ON R.rent_id = P.rent_id
            JOIN users U ON U.user_id = p.user_id;

--List all the events having total sold of more than 100 tickets
SELECT event_name AS EVENT, event_type
      FROM Events
      WHERE event_id IN (SELECT event_id
                                          FROM Event_shows
                                          GROUP BY event_id
                                          HAVING sum(total_seats -
available_seats) > 100
                                );

-- List Event Organizer name and Company name with number of event
show's he organized
SELECT      full_name as Organizer_Name, company_name, COUNT(*) as
Total_shows
      FROM (Event_organizer NATURAL JOIN Users)
      JOIN Company USING(company_id)
      JOIN Event_shows USING(show_id)
      GROUP BY(Organizer_Name, company_name)
      ORDER BY Total_shows DESC;

--List of user who purchased has hightest amount of booking
SELECT  U.full_name, SUM(B.seats * S.price) as Total_payment
      FROM Users U, Bookings B, Event_shows S
      WHERE U.user_id = B.user_id AND B.show_id = S.show_id
      GROUP BY U.user_id
      ORDER BY Total_payment DESC
```



```sql
--List of Customer and Movie name who purchased movies for which they bought tickets as well

SELECT U.full_name as Customer_Name, E.event_name as Event_Name
        FROM events E, event_on_rent R, purchase P, users U
        WHERE R.rent_id = P.rent_id and R.event_id = E.event_id and U.user_id = p.user_id;
```

	customer_name character varying (50)	event_name character varying (100)
1	Margeaux Kitcherside	Faith School Menace?
2	Keely Bantock	Faith School Menace?
3	Deerdre Konert	It's a Great Feeling
4	Daffy Birrane	It's a Great Feeling
5	Dotti Gonsalo	Baran
6	Rudyard Westfield	Waiting for Happiness Heremakono
7	Joel Geffinger	Montana
8	Gay McCobb	Waco The Rules of Engagement
9	Jeff Dowell	Vampire Hunter D Bloodlust Banpaia hantâ D
10	Jenna Alflatt	Curly Top
11	Carolyn Arnaudi	I Live in Fear Ikimono no kiroku
12	Margeaux Kitcherside	Our Children À perdre la raison
13	Joseito Ramsay	Crackerjack
14	Brena Blasdale	Lone Wolf and Cub Baby Cart to Hades Kozure Ôkami...
15	Gregoor MacRierie	Frogmen The
16	Theo Brastead	Alphabet
17	Lynelle Rosendall	Gojoe Spirit War Chronicle Gojo reisenki Gojoe
18	Olimpia Fishbourn	Faith School Menace?
19	Jere Febre	Devil's Own The
20	Vallie Dorrington	Blue Lagoon The
21	Lovdie Christofe	Letter to Elia A

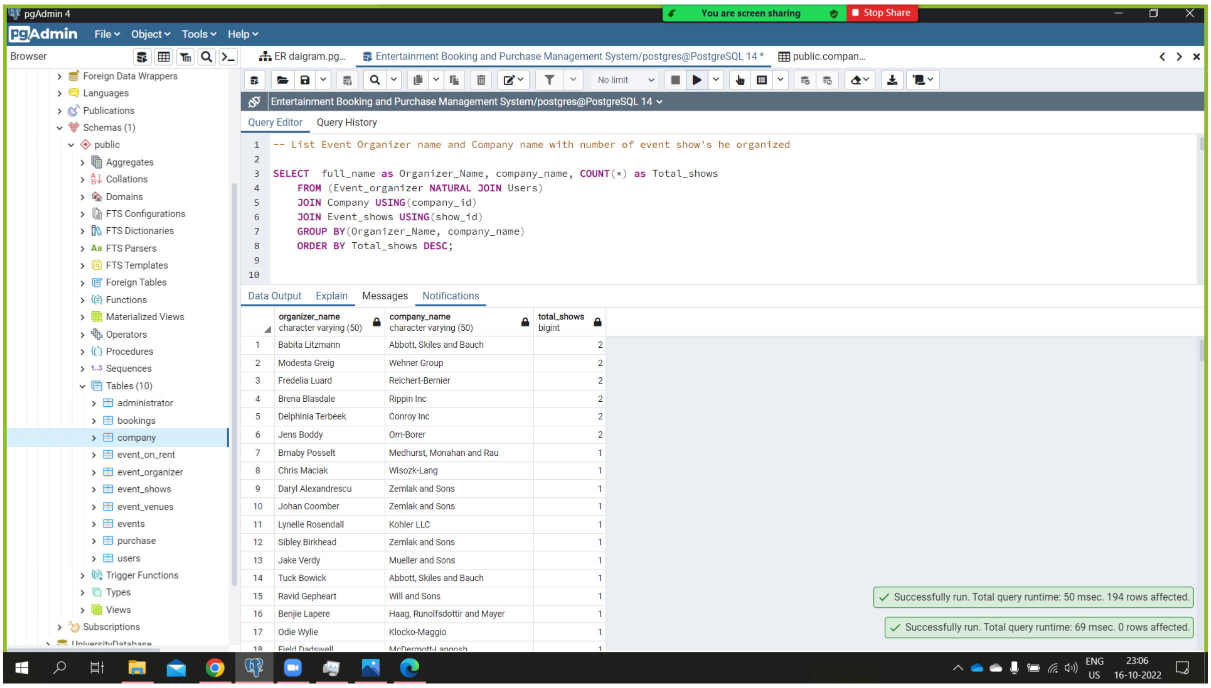
✓ Successfully run. Total query runtime: 46 msec. 400 rows affected.



```sql
--List of Customer and Movie name who purchased movies for which they bought tickets as well

SELECT U.full_name as Customer_Name, E.event_name as Event_Name
        FROM events E
        JOIN event_on_rent R ON R.event_id = E.event_id
        JOIN purchase P ON R.rent_id = P.rent_id
        JOIN users U ON U.user_id = p.user_id;
```

	customer_name character varying (50)	event_name character varying (100)
1	Margeaux Kitcherside	Faith School Menace?
2	Keely Bantock	Faith School Menace?
3	Deerdre Konert	It's a Great Feeling
4	Daffy Birrane	It's a Great Feeling
5	Dotti Gonsalo	Baran
6	Rudyard Westfield	Waiting for Happiness Heremakono
7	Joel Geffinger	Montana
8	Gay McCobb	Waco The Rules of Engagement
9	Jeff Dowell	Vampire Hunter D Bloodlust Banpaia hantâ D
10	Jenna Alflatt	Curly Top
11	Carolyn Arnaudi	I Live in Fear Ikimono no kiroku
12	Margeaux Kitcherside	Our Children À perdre la raison
13	Joseito Ramsay	Crackerjack
14	Brena Blasdale	Lone Wolf and Cub Baby Cart to Hades Kozure Ôkami...
15	Gregoor MacRierie	Frogmen The
16	Theo Brastead	Alphabet
17	Lynelle Rosendall	Gojoe Spirit War Chronicle Gojo reisenki Gojoe
18	Olimpia Fishbourn	Faith School Menace?
19	Jere Febre	Devil's Own The

✓ Successfully run. Total query runtime: 51 msec. 400 rows affected.

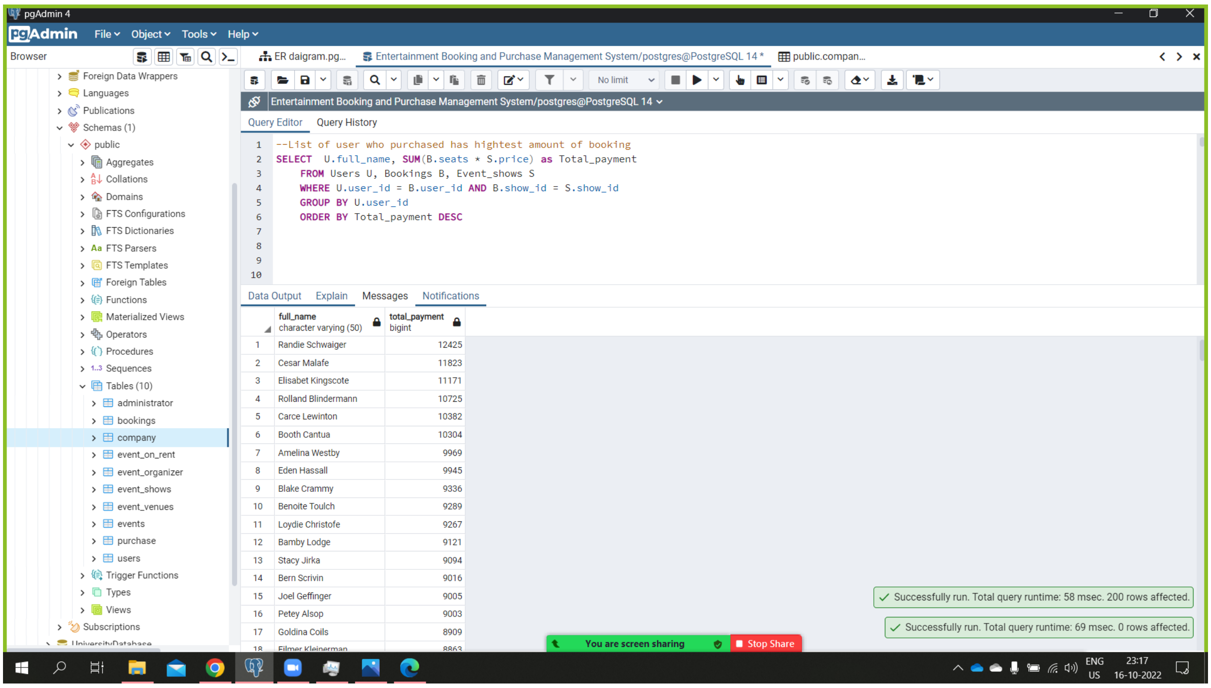


```sql
-- List Event Organizer name and Company name with number of event show's he organized

SELECT  full_name as Organizer_Name, company_name, COUNT(*) as Total_shows
    FROM (Event_organizer NATURAL JOIN Users)
    JOIN Company USING(company_id)
    JOIN Event_shows USING(show_id)
    GROUP BY(Organizer_Name, company_name)
    ORDER BY Total_shows DESC;
```

	organizer_name character varying (50)	company_name character varying (50)	total_shows bigint
1	Babita Litzmann	Abbott, Skiles and Bauch	2
2	Modesta Greig	Wehner Group	2
3	Fredelia Luard	Reichert-Bernier	2
4	Brena Blasdale	Rippin Inc	2
5	Delphinia Terbeek	Conroy Inc	2
6	Jens Boddy	Orn-Borer	2
7	Brnaby Posselt	Medhurst, Monahan and Rau	1
8	Chris Maciak	Wisozk-Lang	1
9	Daryl Alexandrescu	Zemlak and Sons	1
10	Johan Coomber	Zemlak and Sons	1
11	Lynelle Rosendall	Kohler LLC	1
12	Sibley Birkhead	Zemlak and Sons	1
13	Jake Verdy	Mueller and Sons	1
14	Tuck Bowick	Abbott, Skiles and Bauch	1
15	Ravid Gepheart	Will and Sons	1
16	Benjie Lapere	Haag, Runolfsdottir and Mayer	1
17	Odie Wylie	Klocko-Maggio	1
18	Field Dadswell	McDermott-Langosh	1

✓ Successfully run. Total query runtime: 50 msec. 194 rows affected.

✓ Successfully run. Total query runtime: 69 msec. 0 rows affected.



```sql
--List of user who purchased has hightest amount of booking
SELECT  U.full_name, SUM(B.seats * S.price) as Total_payment
    FROM Users U, Bookings B, Event_shows S
    WHERE U.user_id = B.user_id AND B.show_id = S.show_id
    GROUP BY U.user_id
    ORDER BY Total_payment DESC
```

	full_name character varying (50)	total_payment bigint
1	Randie Schwaiger	12425
2	Cesar Malafe	11823
3	Elisabet Kingscote	11171
4	Rolland Blindermann	10725
5	Carce Lewinton	10382
6	Booth Cantua	10304
7	Amelina Westby	9969
8	Eden Hassall	9945
9	Blake Crammy	9336
10	Benoite Toulch	9289
11	Loydie Christofe	9267
12	Bamby Lodge	9121
13	Stacy Jirka	9094
14	Bern Scrivin	9016
15	Joel Geffinger	9005
16	Petey Alsop	9003
17	Goldina Coils	8909
18	Filmer Kleinerman	8863

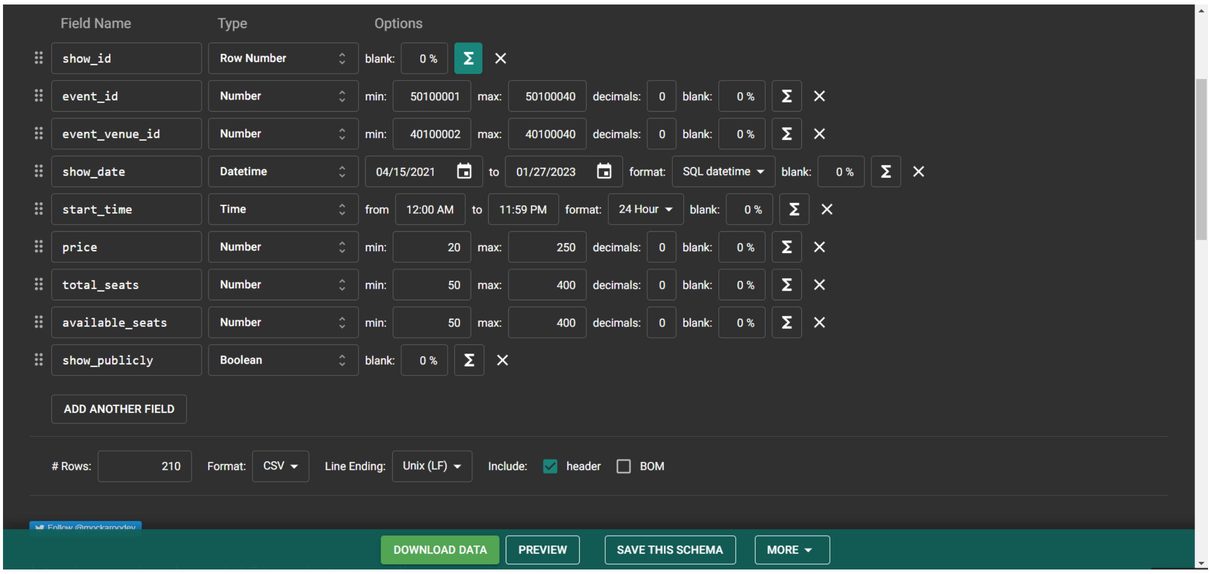
✓ Successfully run. Total query runtime: 58 msec. 200 rows affected.

✓ Successfully run. Total query runtime: 69 msec. 0 rows affected.

7. Data Scraping/ Data import

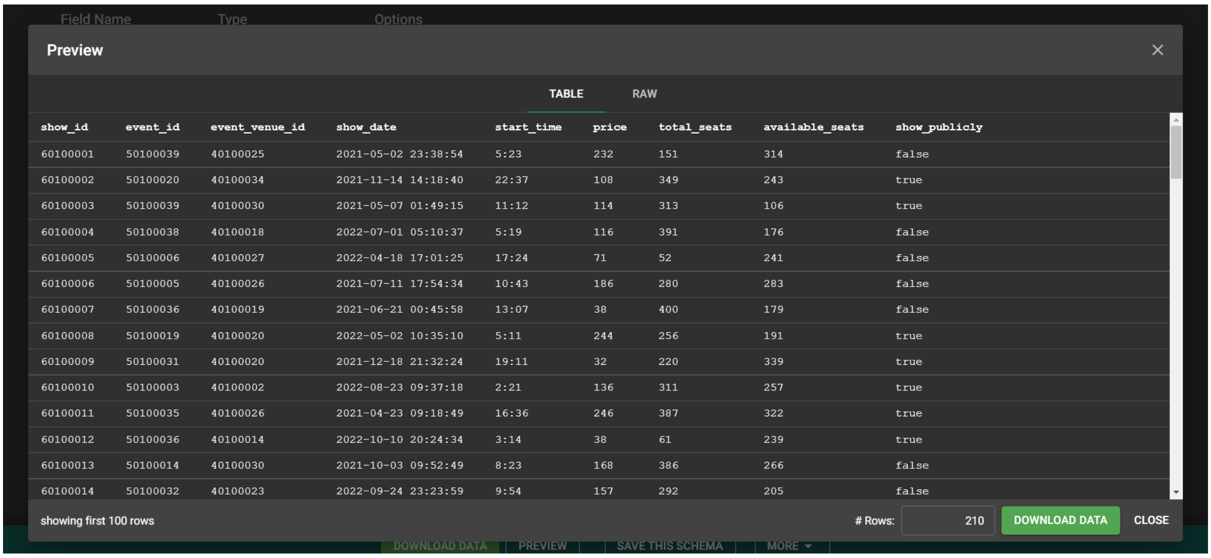
For mock data insert into the database, we use an online tool for data generator https://www.mockaroo.com/ which will generate data based on parameters we provide such as numeric range, date range, custom list, random address/phone, and custom query function, etc.

Below are screenshots of the tool and a preview of the generated data.



Field Name	Type	Options
show_id	Row Number	blank: 0 %
event_id	Number	min: 50100001 max: 50100040 decimals: 0 blank: 0 %
event_venue_id	Number	min: 40100002 max: 40100040 decimals: 0 blank: 0 %
show_date	Datetime	04/15/2021 to 01/27/2023 format: SQL datetime blank: 0 %
start_time	Time	from 12:00 AM to 11:59 PM format: 24 Hour blank: 0 %
price	Number	min: 20 max: 250 decimals: 0 blank: 0 %
total_seats	Number	min: 50 max: 400 decimals: 0 blank: 0 %
available_seats	Number	min: 50 max: 400 decimals: 0 blank: 0 %
show_publicly	Boolean	blank: 0 %

Rows: 210 Format: CSV Line Ending: Unix (LF) Include: ☑ header □ BOM

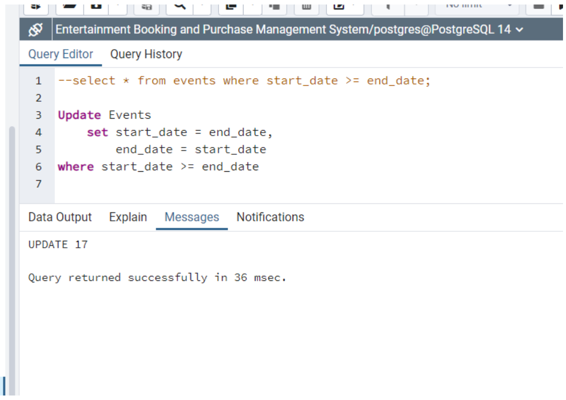


Preview

show_id	event_id	event_venue_id	show_date	start_time	price	total_seats	available_seats	show_publicly
60100001	50100039	40100025	2021-05-02 23:38:54	5:23	232	151	314	false
60100002	50100020	40100034	2021-11-14 14:18:40	22:37	108	349	243	true
60100003	50100039	40100030	2021-05-07 01:49:15	11:12	114	313	106	true
60100004	50100038	40100018	2022-07-01 05:10:37	5:19	116	391	176	false
60100005	50100006	40100027	2022-04-18 17:01:25	17:24	71	52	241	false
60100006	50100005	40100026	2021-07-11 17:54:34	10:43	186	280	283	false
60100007	50100036	40100019	2021-06-21 00:45:58	13:07	38	400	179	false
60100008	50100019	40100020	2022-05-02 10:35:10	5:11	244	256	191	true
60100009	50100031	40100020	2021-12-18 21:32:24	19:11	32	220	339	true
60100010	50100003	40100002	2022-08-23 09:37:18	2:21	136	311	257	true
60100011	50100035	40100026	2021-04-23 09:18:49	16:36	246	387	322	true
60100012	50100036	40100014	2022-10-10 20:24:34	3:14	38	61	239	true
60100013	50100014	40100030	2021-10-03 09:52:49	8:23	168	386	266	false
60100014	50100032	40100023	2022-09-24 23:23:59	9:54	157	292	205	false

showing first 100 rows # Rows: 210

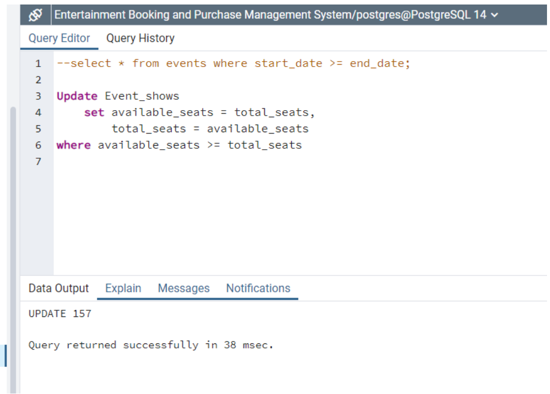
After data generation and importing into Postgres we do some data updation, such as end data should be > start date and available seats should be < total_seats, etc.



Entertainment Booking and Purchase Management System/postgres@PostgreSQL 14

```
--select * from events where start_date >= end_date;

Update Events
    set start_date = end_date,
        end_date = start_date
where start_date >= end_date

```

```
UPDATE 17

Query returned successfully in 36 msec.
```

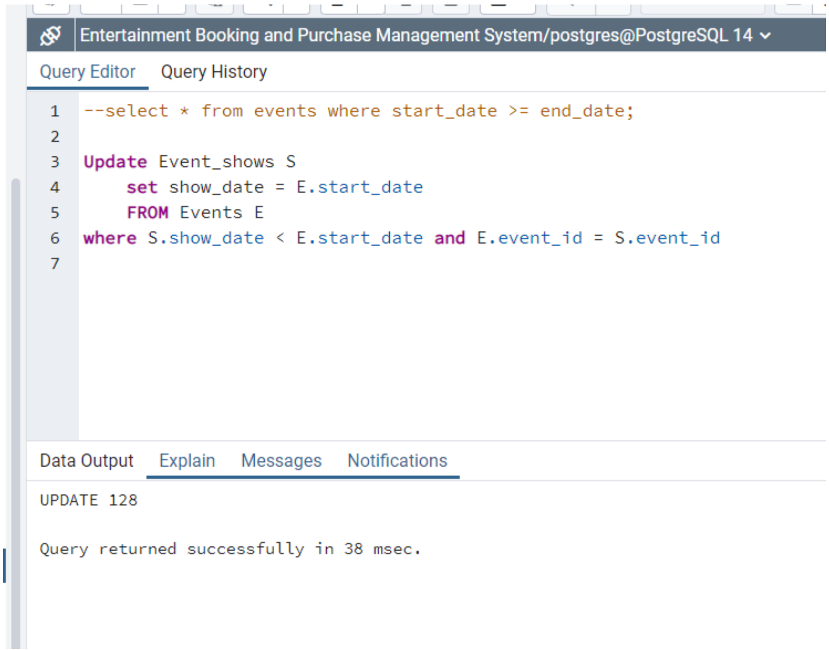


Entertainment Booking and Purchase Management System/postgres@PostgreSQL 14

```
--select * from events where start_date >= end_date;

Update Event_shows
    set available_seats = total_seats,
        total_seats = available_seats
where available_seats >= total_seats

```

```
UPDATE 157

Query returned successfully in 38 msec.
```

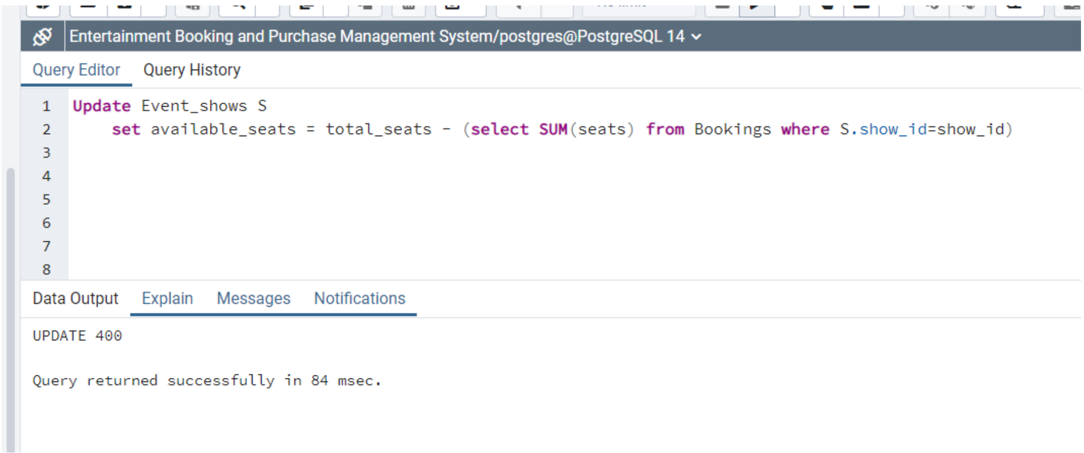
And correct values which are related to other tables, e.g. event_show's start date should >= event's start date


Entertainment Booking and Purchase Management System/postgres@PostgreSQL 14
Query Editor
Query History
Data Output
Explain
Messages
Notifications

```
--select * from events where start_date >= end_date;

Update Event_shows S
    set show_date = E.start_date
    FROM Events E
where S.show_date < E.start_date and E.event_id = S.event_id

```

```
UPDATE 128

Query returned successfully in 38 msec.
```

And No. available seats for the event show should be remaining of the total bookings


Entertainment Booking and Purchase Management System/postgres@PostgreSQL 14
Query Editor
Query History
Data Output
Explain
Messages
Notifications

```
Update Event_shows S
    set available_seats = total_seats - (select SUM(seats) from Bookings where S.show_id=show_id)
```

```
UPDATE 400

Query returned successfully in 84 msec.
```