Information and communication technologies

Industrial Automation

ICT Final Exam, Fall 2020

The information system of booking and buying tickets

Student: Amir Yakubov

Group: IA-2002

Checked by: Samat Kassymkhanov

2020

1. **Define the purpose of your system.**

Short description: My database theme is booking and purchasing tickets. I chose to buy and book tickets in the cinema. Small database system. I have created bases for such as an employee, a customer, a cinema-organization, we can consider it as an administrator, tickets, places and films.

- **What is the purpose of the database? Why is it needed? What should it do?**

The purpose of this base. This is necessary to simplify the use of data: employees, customer list, information, as well as for data analysis. You can find certain information using personal criteria such as ID. Each profile, items have their own id, which are individual, and they do not repeat. You can find information by date or indicated price, it is very convenient.

- **Who are the users and what are their information needs?**

We can consider this system from different points of view, for example, for the administrator, as I said, to get all the information, data about all employees and clients. From the point of view of the employee: he can also get information about the customer, the list is presented and the buying. From the point of view of the client, he can look at the list of films, also what and where are available seats, find the address, mail and phone number to buy or booking a ticket.

- **What are the problems that the system should solve?**

First of all - using this method, you can save a lot of time, and you don't need to exert great effort to achieve what you want.

- **What input data is available to the database?**

First name, last name, email, phone number, address.

- **What kind of information should be stored in the database?**

The database stores such data as names depending on the subject or personality, the names of people or the names of films, somewhere you need to enter both the surname and gender, there is an address, mail, phone numbers. Also the number of the hall, row, seat, prices for these seats. Time of the beginning and the end of the session, genres.

2. Create ERD using Crow's Foot notation (min.10 well-organized entities; their attributes, and types of relations);

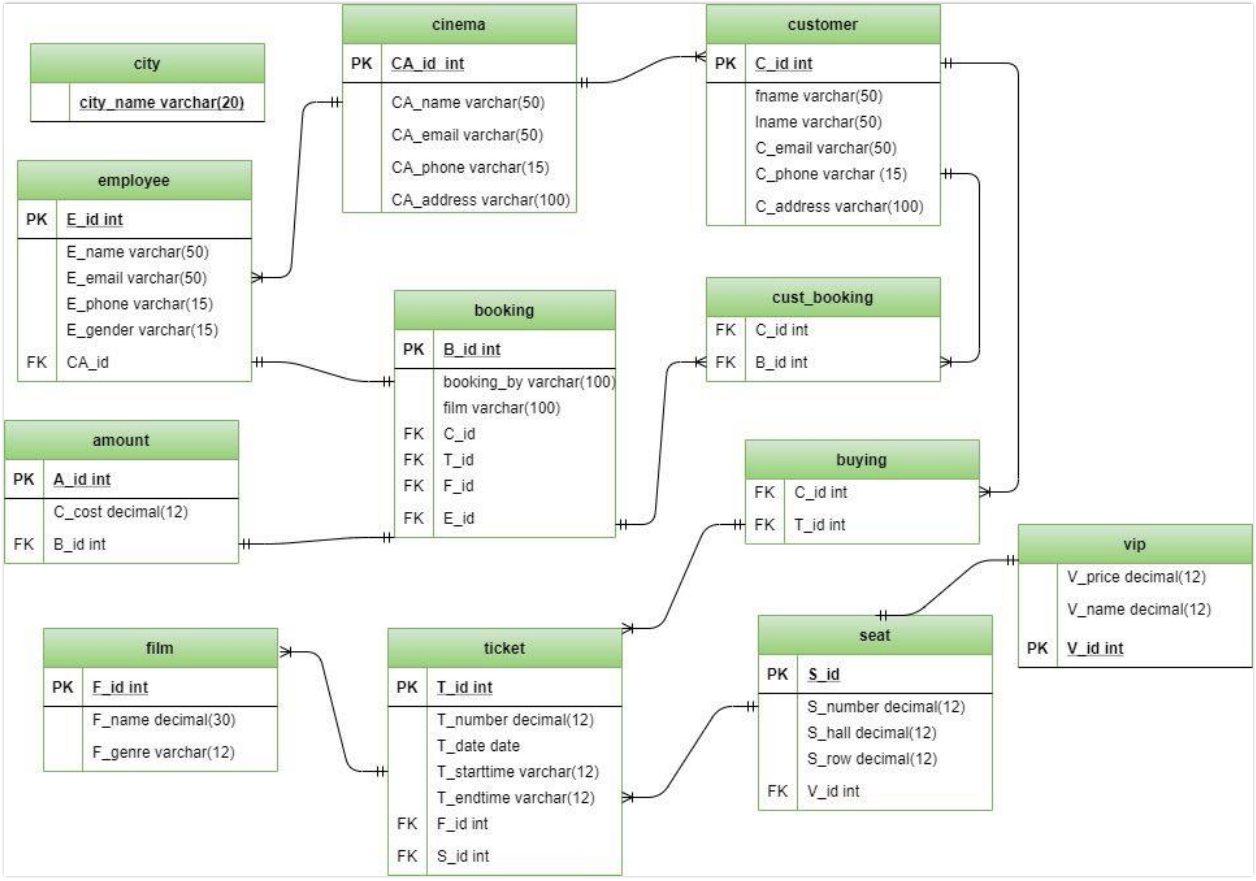

Figure 1 – The ERD diagram of my topic.

Figure 1 shows the tables I created.

Steps of creation.

Step 1. You can see 12 tables here. First, I created a "Cinema" table, which has its own id, mail, name, organization number and address. Then 2 more tables emanate from the cinema, which means that the cinema has two users, so to speak. The first is a "Customer" who has an ID, first name, last name, mail, phone number, and address. And 2 is an "Employee", not much different from the client's data. He also has an ID, name, mail, telephone, and address and is connected to the cinema through the cinema ID.

Step 2. The client can make a reservation. The customer order table links the table itself and the booking table. The "booking" table has data such as which movie was booked for, who booked by, ticket id and movie id.

Step 3. Now the employee, he accepts the customer's order, and here we link to the following table. Which is called the "amount" that stores the total expense of the client.

Step 4. The second thing a client can do is buy a ticket right away, "buying" table links the client and ticket tables.

Step 5. Now the “ticket” table. It has its own id, ticket number, movie date, start and end times. Connects 2 tables - a movie and a seat.

Step 6. Let's start with the “film”. The film also has an id, title and genre.

Step 7. Then the “seat”. A seat has an id, hall number, row number and seat number. It has one table called “vip”.

Step 8. “vip” has a price and a name.

Step 9. And the last table that I did not count is the “City” table, which stores only the name of the city. He is not connected with anything, we need him for the next task, let's move on to him.

3. Create database: tables with entities (tables) and constraints (PK, FK, UK, and etc.);

Tables code:

```
Create table cinema (
CA_Id int primary key,
CA_name varchar(50),
CA_email varchar(50),
CA_phone varchar(15),
CA_address varchar(100));

Create table customer (
C_id int primary key,
fname varchar(50),
lname varchar(50),
C_email varchar(50),
C_phone varchar(15),
C_address varchar(100));

Create table employee (
E_id int primary key,
E_name varchar(50),
E_email varchar(50),
E_phone varchar(15),
E_gender varchar(10),
```

```
CA_id int,
Foreign key(CA_id) references cinema(CA_id));

Create table film (
F_id int primary key,
F_name varchar(30),
F_genre varchar(12));

Create table vip(
V_id int primary key,
V_price decimal(12),
V_name varchar(12));

Create table seat (
S_id int primary key,
S_number decimal(12),
S_hall decimal(12),
S_row decimal(12),
V_id int,
Foreign key (V_id) references vip(V_id));

Create table ticket (
T_id int primary key,
T_number decimal(12),
T_date date,
T_starttime varchar(12),
T_endtime varchar(12),
F_id int,
S_id int,
Foreign key (F_id) references film(F_id),
Foreign key (S_id) references seat(S_id));
Create table buying (
C_id int,
```

```
T_id int,
Foreign key (C_id) references customer(C_id),
Foreign key (T_id) references ticket(T_id));

Create table booking (
B_id int primary key,
booking_by varchar(100),
film varchar(100),
C_id int,
T_id int,
F_id int,
E_id int,
Foreign key (C_id) references customer(C_id),
Foreign key (T_id) references ticket(T_id),
Foreign key (F_id) references film(F_id),
Foreign key (E_id) references employee(E_id));

Create table cust_booking (
C_id int,
B_id int,
Foreign key(C_id) references customer(C_id),
Foreign key(B_id) references booking(B_id));

Create table amount (
A_id int primary key,
C_cost decimal(12),
B_id int,
Foreign key (B_id) references booking(B_id));

Create table city (
city_name varchar(20) primary key);
```

4. Write 5 different (add, drop and constraints) ALTER TABLE statements;

1) The administrator wants to enter a column with the population of the city in order to further calculate how many people visit this cinema.

ALTER TABLE city ADD COLUMN city_population int;

2) The administrator wants to enter that the column with the population of the city visited by this cinema is not null.

ALTER TABLE city ALTER city_population SET not null;

3) The administrator wants to enter that the column with the population of the city that visits this cinema can be null.

ALTER TABLE city alter city_population DROP not null;

4) The administrator wants to delete the city's population column.

ALTER TABLE city DROP COLUMN city_population;

5) The administrator wants to delete the city table itself.

DROP TABLE city;

5. Write SQL query for DML statements (insert, delete, update). Insert - for all tables at least 10 rows, Update – for each table with a condition, Delete – for each table with a condition;

Cinema

Code:

```
INSERT INTO cinema (ca_id, ca_name, ca_email, ca_phone, ca_address)
VALUES (1, 'Star Cinema' , 'astanacinema@mail,ru', '87001010101', 'Nur-Sultan'),
(2, 'Lumiera Cinema', 'almatycin@mail,ru', '87072020202', 'Almaty'),
(12, 'KinoplexX' , 'aktaucin@mail,ru', '87001121212', 'Aktau'),
(13, 'Love Cinema' , 'shymkentcin@mail,ru', '87001131313', 'Shymkent'),
(8, 'Astana Cinema' , 'tarazcin@mail,ru', '87008080808', 'Taraz'),
(14, 'Festival CINEMA' , 'pavlodarcin@mail,ru', '87001141414', 'Pavlodar'),
(11, 'Oiyn Sauyq Ortalygy' , 'kzylordacin@mail,ru', '87001111111', 'Kzyl-Orda'),
(4, 'Локомотив' , 'aktobecin@mail,ru', '87004040404', 'Aktobe'),
(16, 'Алем' , 'semeycin@mail,ru', '87001161616', 'Semey'),
(5, 'Атамекен' , 'taldykorgancin@mail,ru', '87005050505', 'Taldykorgan');
```

Query Editor История запросов

```
1 select*from cinema
2
3
```

Результат План выполнения Сообщения Notifications

	ca_id [PK] integer	ca_name character varying (50)	ca_email character varying (50)	ca_phone character varying (15)	ca_address character varying (100)
1	1	Star Cinema	astanacinema@mail,ru	87001010101	Nur-Sultan
2	2	Lumiera Cinema	almatycin@mail,ru	87072020202	Almaty
3	12	KinoplexX	aktaucin@mail,ru	87001121212	Aktau
4	13	Love Cinema	shymkentcin@mail,ru	87001131313	Shymkent
5	8	Astana Cinema	tarazcin@mail,ru	87008080808	Taraz
6	14	Festival CINEMA	pavlodarcin@mail,ru	87001141414	Pavlodar
7	11	Oiyn Sauyq Ortalygy	kzylordacin@mail,ru	87001111111	Kzyl-Orda
8	4	Локомотив	aktobecin@mail,ru	87004040404	Aktobe
9	16	Алем	semeycin@mail,ru	87001161616	Semey
10	5	Атамекен	taldykorgancin@mail,ru	87005050505	Taldykorgan

Figure 2 - cinema table data.

Figure 2 shows the values and data entered into it.

Customer

Code:

```
insert into customer (c_id, fname, lname, c_email, c_phone, c_address)
values
(01, 'Karly', 'Gallaher', 'kgallaher0@elpais.com', '836-448-0755', '93 Weeping Birch Crossing'),
(02, 'Hildagard', 'Lynam', 'hlynam1@bloomberg.com', '752-307-4672', '80 Merchant Park'),
(03, 'Catharine', 'Holligan', 'cholligan2@trellian.com', '230-287-2593', '701 Prairieview Hill'),
(04, 'Shari', 'Aizikovitz', 'saizikovitz3@biblegateway.com', '801-887-4561', '49436 Twin Pines Terrace'),
(05, 'Archy', 'Paley', 'apaley4@disqus.com', '116-507-7409', '252 Stoughton Alley'),
(06, 'Margareta', 'Arendsen', 'marendsen5@tumblr.com', '747-871-1762', '6 Victoria Point'),
(07, 'Paddie', 'Burgiss', 'pburgiss6@bandcamp.com', '333-915-8139', '5560 Stephen Circle'),
(08, 'Cassius', 'Kempton', 'ckempton7@gnu.org', '997-800-6302', '981 Upham Avenue'),
(09, 'Fredrika', 'Trower', 'ftrower8@tmall.com', '200-273-4132', '44121 3rd Lane'),
(10, 'Lockwood', 'Elsmor', 'lelsmor9@bandcamp.com', '265-778-5659', '72016 Service Junction');
```

Query Editor | История запросов | Scratch Pad

```
select*from customer
```

Результат | План выполнения | Сообщения | Notifications

	c_id [PK] integer	fname character varying (50)	lname character varying (50)	c_email character varying (50)	c_phone character varying (15)	c_address character varying (100)
1	1	Karly	Gallaher	kgallaher0@elpais.com	836-448-0755	93 Weeping Birch Crossing
2	2	Hildagard	Lynam	hlynam1@bloomberg.com	752-307-4672	80 Merchant Park
3	3	Catharine	Holligan	cholligan2@trellian.com	230-287-2593	701 Prairieview Hill
4	4	Shari	Aizikovitz	saizikovitz3@biblegateway.c...	801-887-4561	49436 Twin Pines Terrace
5	5	Archy	Paley	apaley4@disqus.com	116-507-7409	252 Stoughton Alley
6	6	Margareta	Arendsen	marendsen5@tumblr.com	747-871-1762	6 Victoria Point
7	7	Paddie	Burgiss	pburgiss6@bandcamp.com	333-915-8139	5560 Stephen Circle
8	8	Cassius	Kempton	ckempton7@gnu.org	997-800-6302	981 Upham Avenue
9	9	Fredrika	Trower	ftrower8@tmall.com	200-273-4132	44121 3rd Lane
10	10	Lockwood	Elsmor	lelsmor9@bandcamp.com	265-778-5659	72016 Service Junction

Figure 2.1 – customer table data.

Figure 2.1 shows the values and data entered into it.

Employee

Code:

```
insert into employee (e_id, e_name, e_email, e_phone, e_gender, ca_id)
values
(101, 'Moyra', 'mmaron1@dedecms.com', '760-722-9267', 'Female', 2),
(102, 'Hilary', 'hskuce2@tiny.cc', '470-658-3672', 'Male', 8),
(103, 'Petronilla', 'pmackim3@skyrock.com', '387-518-3005', 'Female', 12),
```

(104, 'Idalina', 'ieaston4@pinterest.com', '390-624-8643', 'Female', 13),

(105, 'Cherie', 'ccredland5@1688.com', '357-622-4643', 'Female', 4),

(106, 'Alic', 'abaylie6@mapquest.com', '644-422-4100', 'Male', 5),

(107, 'Abner', 'amottinelli7@businessinsider.com', '595-779-8250', 'Male', 14),

(108, 'Maridel', 'msaunt8@xing.com', '479-834-9002', 'Female', 11),

(109, 'Mead', 'msansbury9@java.com', '439-399-7204', 'Male', 16),

(110, 'Alicia', 'amatveiko0@hc360.com', '367-745-5519', 'Female', 1);

Query Editor История запросов

```
1 select* from employee
2
3
```

Результат План выполнения Сообщения Notifications

	e_id [PK] integer	e_name character varying (50)	e_email character varying (50)	e_phone character varying (15)	e_gender character varying (10)	ca_id integer
1	101	Moyra	mmaron1@dedecms.com	760-722-9267	Female	2
2	102	Hilary	hskuce2@tiny.cc	470-658-3672	Male	8
3	103	Petronilla	pmackim3@skyrock.com	387-518-3005	Female	12
4	104	Idalina	ieaston4@pinterest.com	390-624-8643	Female	13
5	105	Cherie	ccredland5@1688.com	357-622-4643	Female	4
6	106	Alic	abaylie6@mapquest.com	644-422-4100	Male	5
7	107	Abner	amottinelli7@businessinsid...	595-779-8250	Male	14
8	108	Maridel	msaunt8@xing.com	479-834-9002	Female	11
9	109	Mead	msansbury9@java.com	439-399-7204	Male	16
10	110	Alicia	amatveiko0@hc360.com	367-745-5519	Female	1

Figure 2.2 - employee table data.

Figure 2.2 shows the values and data entered into it.

Film

Code:

insert into film (f_id, f_name, f_genre)

values

(31, 'Betsy''s Wedding', 'Comedy'),

(32, 'Steam Experiment, The', 'Drama'),

(33, 'Over the Brooklyn Bridge', 'Comedy'),

(34, 'Haunted House', 'Comedy'),

(35, 'Eye See You', 'Horror'),

(36, 'Public Enemy, The', 'Action'),

(37, 'Man with the Gun', 'Western'),

(38, 'Errors of the Human Body ', '|Thriller'),

(39, 'Kambakkht', 'Action'),

(40, 'Defiant Ones', 'Adventure');

Query Editor История запросов

```
select* from film
```

Результат План выполнения Сообщения Notifications

	f_id [PK] integer	f_name character varying (30)	f_genre character varying (12)
1	31	Betsy's Wedding	Comedy
2	32	Steam Experiment, The	Drama
3	33	Over the Brooklyn Bridge	Comedy
4	34	Haunted House	Comedy
5	35	Eye See You	Horror
6	36	Public Enemy, The	Action
7	37	Man with the Gun	Western
8	38	Errors of the Human Body	|Thriller
9	39	Kambakkht	Action
10	40	Defiant Ones	Adventure

Figure 2.3 - film table data.

Figure 2.3 shows the values and data entered into it.

Vip

Code:

INSERT INTO vip (v_id, v_price, v_name)

VALUES (1, 15000, 'first'),

(2, 10000, 'second'),

(3, 5000, 'third'),

(4, 15000, 'first'),

(5, 5000, 'third'),

(6, 10000, 'second'),

(7, 15000, 'first'),

(8, 5000, 'third'),

(9, 15000, 'first'),

(10, 10000, 'second');

Query Editor История запросов

```
select* from vip
```

Результат План выполнения Сообщения Notifications

	v_id [PK] integer	v_price numeric (12)	v_name character varying (12)
1	1	15000	first
2	2	10000	second
3	3	5000	third
4	4	15000	first
5	5	5000	third
6	6	10000	second
7	7	15000	first
8	8	5000	third
9	9	15000	first
10	10	10000	second

Figure 2.4 - vip table data.

Figure 2.4 shows the values and data entered into it.

Seat

Code:

```
INSERT INTO seat (s_id, s_number, s_hall, s_row, v_id)
VALUES (01, 1, 1, 1, 1),
(02, 2, 1, 2, 2),
(03, 3, 2, 3, 3),
(04, 4, 3, 4, 4),
(05, 5, 2, 5, 5),
(06, 6, 3, 6, 6),
(07, 7, 1, 7, 7),
(08, 8, 1, 8, 8),
(09, 9, 2, 9, 9),
(10, 10, 3, 10, 10);
```

Query Editor История запросов

```
select* from seat
```

Результат План выполнения Сообщения Notifications

	s_id [PK] integer	s_number numeric (12)	s_hall numeric (12)	s_row numeric (12)	v_id integer
1	1	1	1	1	1
2	2	2	1	2	2
3	3	3	2	3	3
4	4	4	3	4	4
5	5	5	2	5	5
6	6	6	3	6	6
7	7	7	1	7	7
8	8	8	1	8	8
9	9	9	2	9	9
10	10	10	3	10	10

Figure 2.5 - seat table data.

Figure 2.5 shows the values and data entered into it.

Ticket

Code:

```
INSERT INTO ticket (t_id, t_number, t_date, t_starttime, t_endtime, f_id, s_id)
VALUES (1001, 2001, '2020-01-01' , '8:00', '9:30', 31, 01),
(1002, 2002, '2020-01-02' , '10:00', '11:30', 32, 02),
(1003, 2003, '2020-01-03' , '15:00', '16:30', 33, 03),
(1004, 2004, '2020-01-02' , '19:00', '21:00', 34, 04),
(1005, 2005, '2020-01-03' , '8:00', '9:30', 35, 05),
(1006, 2006, '2020-01-01' , '10:00', '12:30', 36, 06),
(1007, 2007, '2020-01-01' , '18:00', '19:00', 37, 07),
(1008, 2008, '2020-01-02' , '15:00', '22:30', 38, 08),
(1009, 2009, '2020-01-03' , '22:00', '00:00', 39, 09),
(1010, 2010, '2020-01-01' , '8:00', '9:30', 40, 10);
```

Query Editor История запросов

```
select*from ticket
```

Результат План выполнения Сообщения Notifications

	t_id [PK] integer	t_number numeric (12)	t_date date	t_starttime character varying (12)	t_endtime character varying (12)	f_id integer	s_id integer
1	1001	2001	2020-01-01	8:00	9:30	31	1
2	1002	2002	2020-01-02	10:00	11:30	32	2
3	1003	2003	2020-01-03	15:00	16:30	33	3
4	1004	2004	2020-01-02	19:00	21:00	34	4
5	1005	2005	2020-01-03	8:00	9:30	35	5
6	1006	2006	2020-01-01	10:00	12:30	36	6
7	1007	2007	2020-01-01	18:00	19:00	37	7
8	1008	2008	2020-01-02	15:00	22:30	38	8
9	1009	2009	2020-01-03	22:00	00:00	39	9
10	1010	2010	2020-01-01	8:00	9:30	40	10

Figure 2.6 - ticket table data.

Figure 2.6 shows the values and data entered into it.

Buying

Code:

```
INSERT INTO buying (c_id, t_id)
VALUES (01, 1001),
(02, 1002),
(03, 1003),
(04, 1004),
(05, 1005),
(06, 1006),
(07, 1007),
(08, 1008),
(09, 1009),
(10, 1010);
```

Query Editor История запросов

```
1  select*from buying
2
3
```

Результат План выполнения

	c_id integer	t_id integer
1	1	1001
2	2	1002
3	3	1003
4	4	1004
5	5	1005
6	6	1006
7	7	1007
8	8	1008
9	9	1009
10	10	1010

Figure 2.7 - buying table data.

Figure 2.7 shows the values and data entered into it.

Booking

Code:

INSERT INTO booking (b_id, booking_by, film, c_id, t_id, f_id, e_id)

VALUES (90, 'Karly' , 'Betsy’’s Wedding', 01, 1001, 31, 101),

(91, 'Hildagard' , 'Steam Experiment', 02, 1002, 32, 102),

(92, 'Catherine' , 'Over the Brooklyn Bridge', 03, 1003, 33, 103),

(93, 'Shari' , 'Hounred House', 04, 1004, 34, 104),

(94, 'Archy' , 'Eye See You', 05, 1005, 35, 105),

(95, 'Margareta' , 'Public Enemy', 06, 1006, 36, 106),

(96, 'Paddie' , 'Man With The Gun', 07, 1007, 37, 107),

(97, 'Cassius' , 'Errors of the Human Body', 08, 1008, 38, 108),

(98, 'Fredrika' , 'Kambakht', 09, 1009, 39, 109),

(99, 'Lockwood' , 'Defiant Ones', 10, 1010, 40, 110);

Query Editor История запросов

```
select*from booking
```

Результат План выполнения Сообщения Notifications

	b_id [PK] integer	booking_by character varying (100)	film character varying (100)	c_id integer	t_id integer	f_id integer	e_id integer
1	90	Karly	Betsy"s Wedding	1	1001	31	101
2	91	Hildagard	Steam Experiment	2	1002	32	102
3	92	Catherine	Over the Brooklyn Bridge	3	1003	33	103
4	93	Shari	Hounred House	4	1004	34	104
5	94	Archy	Eye See You	5	1005	35	105
6	95	Margareta	Public Enemy	6	1006	36	106
7	96	Paddie	Man With The Gun	7	1007	37	107
8	97	Cassius	Errors of the Human Body	8	1008	38	108
9	98	Fredrika	Kambakht	9	1009	39	109
10	99	Lockwood	Defiant Ones	10	1010	40	110

Figure 2.8 - booking table data.

Figure 2.8 shows the values and data entered into it.

Cust_booking

Code:

```
INSERT INTO cust_booking(c_id, b_id)
VALUES (01, 91),
(02, 92),
(03, 93),
(04, 94),
(05, 95),
(06, 96),
(07, 97),
(08, 98),
(09, 99),
(10, 90);
```

Query Editor История запросов

```
1  select*from cust_booking
2
3
```

Результат План выполнения Сообщения Notifications

	c_id integer	b_id integer
1	1	91
2	2	92
3	3	93
4	4	94
5	5	95
6	6	96
7	7	97
8	8	98
9	9	99
10	10	90

Figure 2.9 – cust_buying table data.

Figure 2.9 shows the values and data entered into it.

Amount

Code:

```
INSERT INTO amount (a_id, c_cost, b_id)
VALUES (301, 5675, 91),
(302, 6500, 90),
(303, 10000, 92),
(304, 12365, 93),
(305, 15000, 94),
(306, 8700, 95),
(307, 4505, 96),
(308, 7854, 97),
(309, 9000, 98),
(310, 12375, 99);
```

Query Editor История запросов

```
1  select*from amount
2
```

Результат План выполнения Сообщения Notifications

	a_id [PK] integer	c_cost numeric (12)	b_id integer
1	301	5675	91
2	302	6500	90
3	303	10000	92
4	304	12365	93
5	305	15000	94
6	306	8700	95
7	307	4505	96
8	308	7854	97
9	309	9000	98
10	310	12375	99

Figure 2.10 - amount table data.

Figure 2.10 shows the values and data entered into it.

Updates.

1) I give the command: set the amount 14505, where the id amount is 307.

update amount

set c_cost = 14505

where a_id = 307;

Query Editor История запросов

```
1  select c_cost from amount
2  where a_id = 307;
```

Результат План выполнения Сообщения Notifications

	c_cost numeric (12)
1	14505

Figure 3 – update table

Figure 3 shows table change

2) I give the command: set the film Avatar, where the b_id is 98.

update booking

set film = 'Avatar'

where b_id = 98;

Query Editor История запросов

```
1  select film from booking
2  where b_id =98;
```

Результат План выполнения Сообщения Notifications

	film character varying (100)
1	Avatar

Figure 3.1 – update table

Figure 3.1 shows table change

3) I give the command: set t_id 1010, where the c_id more than 9.

update buying

set t_id = 1010

where c_id > 9

Query Editor История запросов

```
1  select t_id
2  from buying
3  where c_id > 9
```

Результат План выполнения Сообщения Notifications

	t_id integer
1	1010

Figure 3.2 – update table

Figure 3.2 shows table change

4) I give the command: set cinema name Kinoplex, where the cinema address is Aktau.

update cinema

set ca_name = 'Kinoplex'

where ca_address = 'Aktau'

Query Editor История запросов

```
1  select ca_name
2  from cinema
3  where ca_address = 'Aktau'
```

Результат План выполнения Сообщения Notifications

	ca_name character varying (50)
1	Kinoplex

Figure 3.3 – update table

Figure 3.3 shows table change

5) I give the command: set b_id 91, where the c_id less than 2.

update cust_booking

set b_id = 91

where c_id < 2

```
5  select b_id
6  from cust_booking
7  where c_id < 2
```

Результат План выполнения Сообщения Notifications

	b_id integer
1	91

Figure 3.4 – update table

Figure 3.4 shows table change

6) I give the command: set first name Karl, where the c_id is 1.

update customer

set fname = 'Karl'

where c_id = 1

```
5 select fname
6 from customer
7 where c_id = 1
```

Результат | План выполнения | Сообщения | Notifications

	fname character varying (50)
1	Karl

Figure 3.5 – update table

Figure 3.5 shows table change

7) I give the command: set employee name Alisa, where the c_id is 5.

update employee

set e_name = 'Alisa'

where ca_id = 5

```
5 select e_name
6 from employee
7 where ca_id = 5
```

Результат | План выполнения | Сообщения | Notifications

	e_name character varying (50)
1	Alisa

Figure 3.6 – update table

Figure 3.6 shows table change

8) I give the command: set film name Avatar, where the f_id 39.

update film

set f_name = 'Avatar'

where f_id = 39

```
5 select f_name
6 from film
7 where f_id = 39
```

Результат | План выполнения | Сообщения | Notifications

	f_name character varying (30)
1	Avatar

Figure 3.7 – update table

Figure 3.7 shows table change

9) I give the command: set hall 1, where the s_id 2.

update seat

set s_hall = 1

where s_id = 2

```
5 select s_hall
6 from seat
7 where s_id = 2
```

Результат | План выполнения | Сообщения | Notifications

	s_hall numeric (12)
1	1

Figure 3.8 – update table

Figure 3.8 shows table change

10) I give the command: set end time 10:00, where the t_id is 1005.

update ticket

set t_endtime = '10:00'

where t_id = 1005

```
5  select t_endtime
6  from ticket
7  where t_id = 1005
```

Результат | План выполнения | Сообщения | Notifications

	t_endtime character varying (12)
1	10:00

Figure 3.9 – update table

Figure 3.9 shows table change

11) I give the command: set vip price 15005, where the v_id is first.

update vip

set v_price = 15005

where v_name = 'first'

```
5  select v_price
6  from vip
7  where v_name = 'first'
```

Результат | План выполнения | Сообщения | Notifications

	v_price numeric (12)
1	15005
2	15005
3	15005
4	15005

Figure 3.10 – update table

Figure 3.10 shows table change

Delete. Delete data fulfilling the specified conditions.

Codes:

delete from amount

where a_id = 301

delete from booking

where b_id = 1

delete from cust_booking

where b_id = 98

delete from amount

where c_cost = 9000

delete from booking

where film = 'Avatar'

delete from employee

where ca_id = 16

delete from cinema

where ca_address = 'Semey'

6. Write at least 10 queries: using DISTINCT, conditions (<, >, =), OR, AND, BETWEEN, IN, LIKE, LENGHT, COUNT, MAX, MIN, SUM, AVG, INNER JOIN, LEFT JOIN, RIGHT JOIN, FULL JOIN and etc. The queries should be coherent and complex.

1) Select employee name where name start with letter A.

Code:

```
Select e_name from employee
Where e_name like 'A%';
```

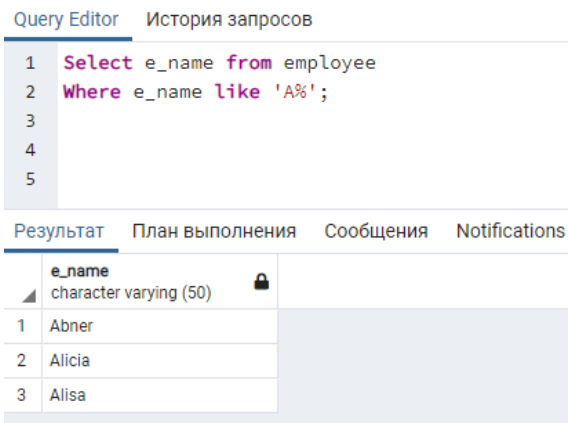

Figure 4 - select data

Figure 4 shows what we want to choose

2) <u>Select total cost of customers</u>

Code:

Select sum(C_cost)

From amount;

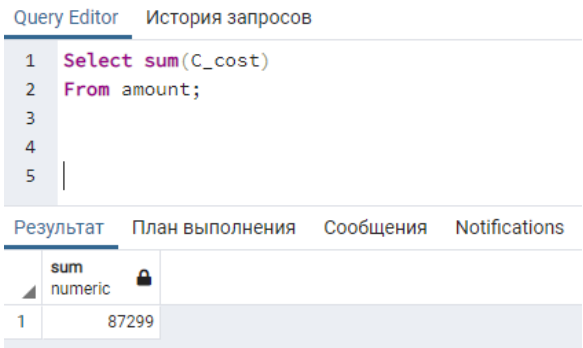

Figure 4.1 - select data

Figure 4.1 shows what we want to choose

3) <u>Select film name Avatar</u>

Code:

Select * from film

Where F_name = 'Avatar';

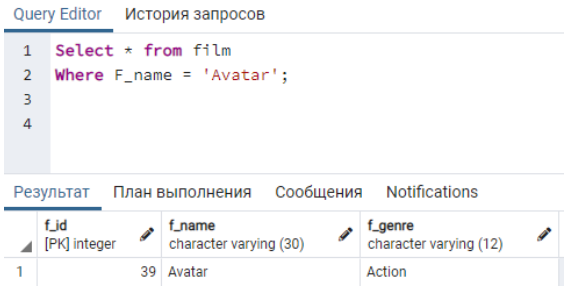

Figure 4.2 - select data

Figure 4.2 shows what we want to choose

4) <u>Select film name where time bettwen 15 and 19</u>

Code:

Select F_name

From film

inner join ticket

on ticket.F_id = film.F_id

Where ticket.t_starttime > '15:00' and ticket.t_endtime < '19:00';

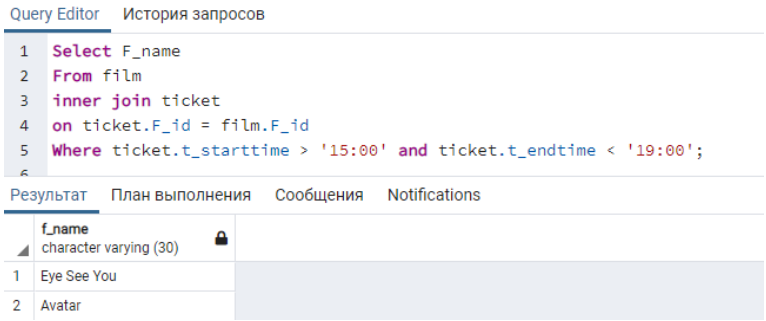

Figure 4.3 - select data

Figure 4.3 shows what we want to choose

5) <u>Select count of first</u>

Code:

select count(distinct 'first') as "the most expensive"

from vip;

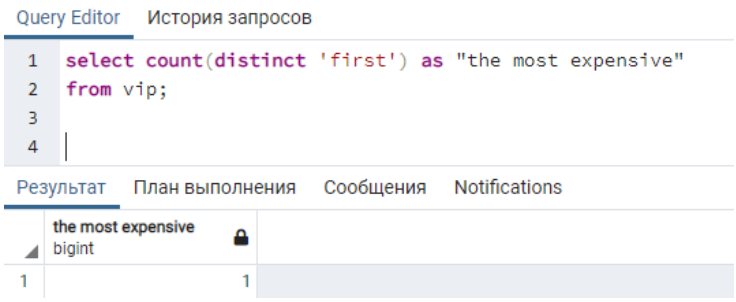

Figure 4.4 - select data

Figure 4.4 shows what we want to choose

6) <u>Select employee name, email and cinema name,email</u>

Code:

```
select E_name, E_email, CA_name, CA_email
from employee
inner join cinema
on cinema.CA_id=employee.CA_id
```

Query Editor История запросов

```
select E_name, E_email, CA_name, CA_email
from employee
inner join cinema
on cinema.CA_id=employee.CA_id
```

Результат Сообщения План выполнения Notifications

	e_name character varying (50)	e_email character varying (50)	ca_name character varying (50)	ca_email character varying (50)
1	Moyra	mmaron1@dedecms.com	Lumiera Cinema	almatycin@mail,ru
2	Hilary	hskuce2@tiny.cc	Astana Cinema	tarazcin@mail,ru
3	Petronilla	pmackim3@skyrock.com	Kinoplex	aktaucin@mail,ru
4	Idalina	ieaston4@pinterest.com	Love Cinema	shymkentcin@mail,ru
5	Cherie	ccredland5@1688.com	Локомотив	aktobecin@mail,ru
6	Abner	amottinelli7@businessinsid...	Festival CINEMA	pavlodarcin@mail,ru
7	Maridel	msaunt8@xing.com	Oiyn Sauyq Ortalygy	kzylordacin@mail,ru
8	Alicia	amatveiko0@hc360.com	Star Cinema	astanacinema@mail,ru
9	Alisa	abaylie6@mapquest.com	Атамекен	taldykorgancin@mail,ru

Figure 4.5 - select data

Figure 4.5 shows what we want to choose

7) <u>Select vip id, price, name where name = third</u>

Code:

```
Select V_id, V_price, V_name
From vip
Where V_name IN ('third');
```

Query Editor История запросов

```
Select V_id, V_price, V_name
From vip
Where V_name IN ('third');
```

Результат Сообщения План выполнения Notifications

	v_id [PK] integer	v_price numeric (12)	v_name character varying (12)
1	3	5000	third
2	5	5000	third
3	8	5000	third

Figure 4.6 - select data

Figure 4.6 shows what we want to choose

8) Select hall, row numbers order by hall descending

Code:

Select s_hall, s_row

From seat

Order by s_hall desc;

Query Editor История запросов

```
Select s_hall, s_row
From seat
Order by s_hall desc;
```

Результат Сообщения План выполнения Notifications

	s_hall numeric (12)	s_row numeric (12)
1	1	2
2	1	7
3	1	8
4	1	1
5	2	9
6	2	3
7	2	5
8	3	6
9	3	10
10	3	4

Figure 4.7 - select data

Figure 4.7 shows what we want to choose

9) <u>Select average of vip price</u>

Code:

Select round(avg(V_price))

From vip;

Query Editor История запросов

```
1  Select round(avg(V_price))
2  From vip;
3
4
```

Результат Сообщения План выполнения Notifications

	round numeric
1	10502

Figure 4.8 - select data

Figure 4.8 shows what we want to choose

10) Select cinema id, name, email, phone, address where id more than 3

Code:

Select ca_id, ca_name, ca_email, ca_phone, ca_address

From cinema

Group by ca_name, ca_email, ca_phone, ca_address, ca_id

Having (ca_id> 3);

Query Editor История запросов Scr

```
1  Select ca_id, ca_name, ca_email, ca_phone, ca_address
2  From cinema
3  Group by ca_name, ca_email, ca_phone, ca_address, ca_id
4  Having (ca_id> 3);
```

Результат Сообщения План выполнения Notifications

	ca_id [PK] integer	ca_name character varying (50)	ca_email character varying (50)	ca_phone character varying (15)	ca_address character varying (100)
1	4	Локомотив	aktobecin@mail,ru	87004040404	Aktobe
2	12	Kinoplex	aktaucin@mail,ru	87001121212	Aktau
3	14	Festival CINEMA	pavlodarcin@mail,ru	87001141414	Pavlodar
4	13	Love Cinema	shymkentcin@mail,ru	87001131313	Shymkent
5	5	Атамекен	taldykorgancin@mail,ru	87005050505	Taldykorgan
6	11	Oiyn Sauyq Ortalygy	kzylordacin@mail,ru	87001111111	Kzyl-Orda
7	8	Astana Cinema	tarazcin@mail,ru	87008080808	Taraz

Figure 4.9 - select data

Figure 4.9 shows what we want to choose

7. Write at least 5 subqueries: single-row, multiple-row and multiple-column subqueries, and etc.;

1) <u>Single column</u>

Code:

Select ca_address

From cinema

Query Editor История запросов

```
Select ca_address
From cinema
```

Результат Сообщения План выполнения Notifications

	ca_address character varying (100)
1	Nur-Sultan
2	Almaty
3	Shymkent
4	Taraz
5	Pavlodar
6	Kzyl-Orda
7	Aktobe
8	Taldykorgan
9	Aktau

Figure 5 – single column

Figure 5 shows single column

2) <u>Single row</u>

Code:

Select * from customer

where c_id =1;

Query Editor История запросов Scratch Pad

```
Select * from customer
where c_id =1;
```

Результат Сообщения План выполнения Notifications

	c_id [PK] integer	fname character varying (50)	lname character varying (50)	c_email character varying (50)	c_phone character varying (15)	c_address character varying (100)
1	1	Karl	Gallaher	kgallaher0@elpais.com	836-448-0755	93 Weeping Birch Crossing

Figure 5.1 – single row

Figure 5.1 shows single row

3) <u>Multiple column</u>

Code:

Select e_name, e_email, e_gender

from employee

Query Editor История запросов

```
Select e_name, e_email, e_gender
from employee
```

Результат Сообщения План выполнения Notifications

	e_name character varying (50)	e_email character varying (50)	e_gender character varying (10)
1	Moyra	mmaron1@dedecms.com	Female
2	Hilary	hskuce2@tiny.cc	Male
3	Petronilla	pmackim3@skyrock.com	Female
4	Idalina	ieaston4@pinterest.com	Female
5	Cherie	ccredland5@1688.com	Female
6	Abner	amottinelli7@businessinsid...	Male
7	Maridel	msaunt8@xing.com	Female
8	Alicia	amatveiko0@hc360.com	Female
9	Alisa	abaylie6@mapquest.com	Male

Figure 5.2 – multiple column

Figure 5.1 shows multiple column

4) <u>Multiple row</u>

Code:

Select f_name, f_genre

from film

where f_id = 33 or f_id = 37 or f_id = 39

Query Editor История запросов

```
Select f_name, f_genre
from film
where f_id = 33 or f_id = 37 or f_id = 39
```

Результат Сообщения План выполнения Notifications

	f_name character varying (30)	f_genre character varying (12)
1	Over the Brooklyn Bridge	Comedy
2	Man with the Gun	Western
3	Avatar	Action

Figure 5.3 – multiple row

Figure 5.1 shows multiple row

5) <u>All columns and rows</u>

Code:

select*from ticket

Query Editor История запросов

```
select*from ticket
```

Результат Сообщения План выполнения Notifications

	t_id [PK] integer	t_number numeric (12)	t_date date	t_starttime character varying (12)	t_endtime character varying (12)	f_id integer	s_id integer
1	1001	2001	2020-01-01	8:00	9:30	31	1
2	1002	2002	2020-01-02	10:00	11:30	32	2
3	1003	2003	2020-01-03	15:00	16:30	33	3
4	1004	2004	2020-01-02	19:00	21:00	34	4
5	1006	2006	2020-01-01	10:00	12:30	36	6
6	1007	2007	2020-01-01	18:00	19:00	37	7
7	1008	2008	2020-01-02	15:00	22:30	38	8
8	1009	2009	2020-01-03	22:00	00:00	39	9
9	1010	2010	2020-01-01	8:00	9:30	40	10
10	1005	2005	2020-01-03	8:00	10:00	35	5

Figure 5.4 – all columns and rows

Figure 5.4 shows all columns and rows

Appendix

Cinema

Primary key: CA_Id

CA_name varchar(50) - must not exceed 50 characters, else error

CA_email varchar(50) - must not exceed 50 characters, else error

CA_phone varchar(15) - must not exceed 15 characters, else error

CA_address varchar(100)) - must not exceed 100 characters, else error

Customer

Primary key: C_id

fname varchar(50) - must not exceed 50 characters, else error

lname varchar(50) - must not exceed 50 characters, else error

C_email varchar(50) - must not exceed 50 characters, else error

C_phone varchar(15) - must not exceed 15 characters, else error

C_address varchar(100) - must not exceed 100 characters, else error

Employee

Primary key: E_id

E_name varchar(50) - must not exceed 50 characters, else error

E_email varchar(50) - must not exceed 50 characters, else error

E_phone varchar(15) - must not exceed 15 characters, else error

E_gender varchar(10) - must not exceed 10 characters, else error

Foreign key: CA_id int

Film

Primary key: F_id

F_name varchar(30) - must not exceed 30 characters, else error

F_genre varchar(12) - must not exceed 12 characters, else error

Vip

Primary key: V_id

V_price decimal(12) - must not exceed 12 numbers, else error

V_name varchar(12) - must not exceed 12 characters, else error

Seat

Primary key: S_id

S_number decimal(12) - must not exceed 12 numbers, else error

S_hall decimal(12) - must not exceed 12 numbers, else error

S_row decimal(12) - must not exceed 12 numbers, else error

Foreign key: V_id

Ticket

Primary key: T_id

T_number decimal(12) - must not exceed 12 numbers, else error

T_date date – format “year-month-day”

T_starttime varchar(12) - must not exceed 12 characters, else error

T_endtime varchar(12) - - must not exceed 12 characters, else error

Foreign key: F_id

Foreign key: S_id

Buying

Foreign key: C_id

Foreign key: T_id

Booking

Primary key: B_id

booking_by varchar(100) - must not exceed 100 characters, else error

film varchar(100) - must not exceed 100 characters, else error

Foreign key: C_id

Foreign key: T_id

Foreign key: F_id

Foreign key: E_id

Cust_booking

Foreign key: C_id

Foreign key: B_id

Amount

Primary key: A_id

C_cost decimal(12) - must not exceed 12 numbers, else error

Foreign key: B_id

Conclusion

In conclusion, I want to say that during the time I did this project, I repeated and maybe even learned a lot of things. I repeated the languages DDL and DML. Found some info about the cinema system and other stuff. Came up with some data. I worked on this project completely alone. So I spent a lot of my time doing this assignment. I think in a group of at least 2 people it would be much faster and even easier. So, this is what I did.

Reference List

[1] Aitmukhanbetova.E, SQL DEVELOPMENT, DDL STATEMENTS from http://moodle.astanait.edu.kz/mod/resource/view.php?id=8226

[2] Aitmukhanbetova.E, SQL DEVELOPMENT, DML STATEMENTS from http://moodle.astanait.edu.kz/mod/resource/view.php?id=8469

[3] Aitmukhanbetova.E, JOIN OPERATORS from http://moodle.astanait.edu.kz/mod/resource/view.php?id=8798

[4] INSERT from https://www.mockaroo.com/