Author: Alimbaeva Burulai

University: Óbudai Egyetem – Neumann János Informatikai Kar

Major: Computer Engineering BSc

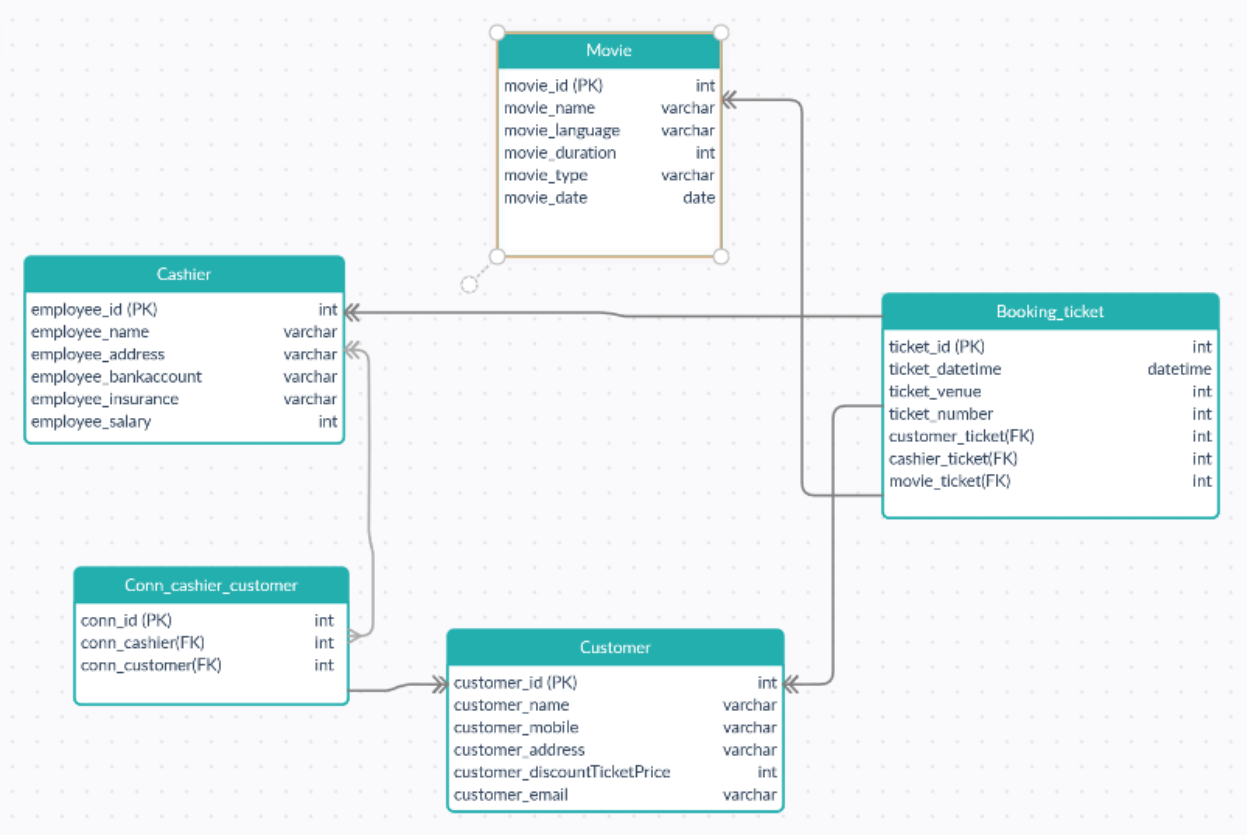
Topic: Movie ticketing system

DATABASE

Description. Movie ticketing system database is aimed to provide complete information about movie and schedule to the customer, according to which she/he can easily get the ticket, can book her/his favorite films, can reserve a seat and can order and get the ticket online as well. Admin can use movie ticketing system to add, delete and update the data such as movie and its description and schedule which is accessible for customers.

The table structure diagram and ER diagram of database can be seen in Figure 1. and 2. below.

Figure 1.



ER diagram of database

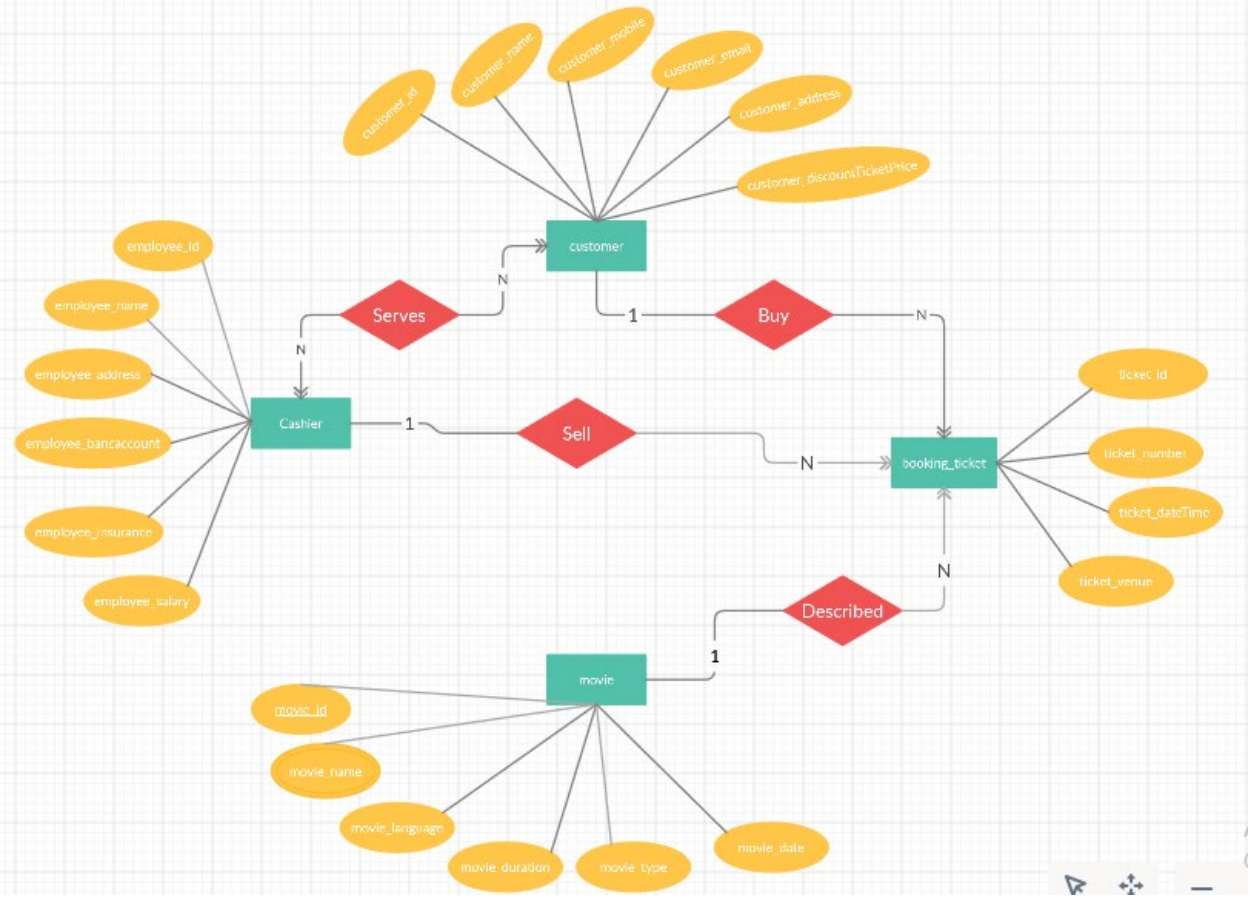


Figure 2.

Rules of Normalization.

Normalization: is a process of organizing the data in database to avoid data redundancy, insertion anomaly, update and deletion anomaly.

First Normal form: an attribute (column) of a table cannot hold multiple values. It should hold only single values.

Second Normal Form: table is in 1NF. Every filed is functionally dependent on the key of table.

Third Normal Form: table must be in 2NF. And it should not have Transitive Dependency. Transitive dependency: when a non-prime attribute depends on other non-prime attributes rather than depending upon the prime attributes or primary key.

BNCF: A table structure is said to be BCNF if it is in 3NF and there are no inner functional dependencies between any of the complex keys' field.

Table of ticketing system

Normalization :0NF

customer_id	customer_name	customer_mobile	customer_email	customer_address	customer_discounte dTicketPrice
1	Alimbaeva Burulai	0 770 486 548	burulai@gmail.com	Csengery 84	1500 HUF
2	Abdulrahman Nuha	0 553 365 211	nuha@gmail.com	Becsi 86	2000HUF
3	Manas kyzy Begai	0 778 543 987	begai@gmail.com	Istvan 65	
4	Hedl Petra	0 556 789 421	petra@gmail.com	Izabella 78	1700 HUF
5	Barta Zoltan Kevin	0 774 564 345	kevin@gmail.com	Nefelejcs 67	

- 1NF is true, because every column holds only single values.

1NF

customer_id	customer_name	customer_mobile	customer_email	customer_address	customer_discounte dTicketPrice
1	Alimbaeva Burulai	0 770 486 548	burulai@gmail.com	Csengery 84	1500 HUF
2	Abdulrahman Nuha	0 553 365 211	nuha@gmail.com	Becsi 86	2000HUF
3	Manas kyzy Begai	0 778 543 987	begai@gmail.com	Istvan 65	2000HUF
4	Hedl Petra	0 556 789 421	petra@gmail.com	Izabella 78	1700 HUF
5	Barta Zoltan Kevin	0 774 564 345	kevin@gmail.com	Nefelejcs 67	1700HUF

- 2NF is true, because every column, which is not primary key, is fully dependent to primary key.

2NF

F custimer_id -->name,mobile,email,address,discountedPrice

customer_id	customer_name	customer_mobile	customer_email	customer_address	customer_discounte dTicketPrice
1	Alimbaeva Burulai	0 770 486 548	burulai@gmail.com	Csengery 84	1500 HUF
2	Abdulrahman Nuha	0 553 365 211	nuha@gmail.com	Becsi 86	2000HUF
3	Manas kyzy Begai	0 778 543 987	begai@gmail.com	Istvan 65	2000HUF
4	Hedl Petra	0 556 789 421	petra@gmail.com	Izabella 78	1700 HUF
5	Barta Zoltan Kevin	0 774 564 345	kevin@gmail.com	Nefelejcs 67	1700HUF

- 3NF is true, because there is no non-key filed that determines another attribute.
- BNCF is true, because the table structure does not contain any complex keys.

Lossless decomposition

Decomposition is lossless if it is feasible to reconstruct relation R from decomposed tables using Joins. This is the preferred choice. The information will not lose from the relation when decomposed.

Base state of database. STEP 1

	cashier_id	cashier_name	cashier_address	cashier_bankaccount	cashier_insurance	cashier_salary	movie_id	movie_name	movie_language	movie_duration	movie_type	movie_date	customer_id	customer_name	customer_mobile
Cashier	B(1,1)	B(1,2)	B(1,3)	B(1,4)	B(1,5)	B(1,6)	B(1,7)	B(1,8)	B(1,9)	B(1,10)	B(1,11)	B(1,12)	B(1,13)	B(1,14)	B(1,15
Movie	B(2,1)	B(2,2)	B(2,3)	B(2,4)	B(2,5)	B(2,6)	B(2,7)	B(2,8)	B(2,9)	B(2,10)	B(2,11)	B(2,12)	B(2,13)	B(2,14)	B(2,15
Customer	B(3,1)	B(3,2)	B(3,3)	B(3,4)	B(3,5)	B(3,6)	B(3,7)	B(3,8)	B(3,9)	B(3,10)	B(3,11)	B(3,12)	B(3,13)	B(3,14)	B(3,15
Booking ticket	B(4,1)	B(4,2)	B(4,3)	B(4,4)	B(4,5)	B(4,6)	B(4,7)	B(4,8)	B(4,9)	B(4,10)	B(4,11)	B(4,12)	B(4,13)	B(4,14)	B(4,15
Conn_cashier_customer	B(5,1)	B(5,2)	B(5,3)	B(5,4)	B(5,5)	B(5,6)	B(5,7)	B(5,8)	B(5,9)	B(5,10)	B(5,11)	B(5,12)	B(5,13)	B(5,14)	B(5,15

DATABASE

STEP 2. Functional dependencies.

Functional Dependencies. STEP 2	
Cashier	$FD_{cashier}$:{cashier_id} --->{cashier_name,cashier_address,cashier_bancaccount,cashier_insurance,cashier_salary}
Movie	FD_{movie}:{movie_id} --->{movie_name,cashier_address,cashier_bancaccount,cashier_insurance,cashier_salary}
Customer	$FD_{customer}$:{customer_id} --->{customer_name,customer_mobile,customer_email,customer_address,customer_discountTicketPrice}
Booking ticket	FD_{ticket}:{ticket_id} --->{ticket_dateTime,ticket_venue,ticket_number,customer_ticket(**FK** - ref : customer_id),cashier_ticket(**FK** - ref : cashier
Conn_cashier_customer	$Fd_{conn\_cashier\_customer}$:{conn_id} --->{conn_cashier(ref : cashier_id),conn_customer(ref : customer_id)}

STEP 3. Initiating the chaser algorithm.

Initiating the chaser algorithm. STEP 3															
Cashier	A(1)	A(2)	A(3)	A(4)	A(5)	A(6)	B(1,7)	B(1,8)	B(1,9)	B(1,10)	B(1,11)	B(1,12)	B(1,13)	B(1,14)	B(1,15)
Movie	B(2,1)	B(2,2)	B(2,3)	B(2,4)	B(2,5)	B(2,6)	A(7)	A(8)	A(9)	A(10)	A(11)	A(12)	B(2,13)	B(2,14)	B(2,15)
Customer	B(3,1)	B(3,2)	B(3,3)	B(3,4)	B(3,5)	B(3,6)	B(3,7)	B(3,8)	B(3,9)	B(3,10)	B(3,11)	B(3,12)	A(13)	A(14)	A(15)
Booking ticket	A(1)	B(4,2)	B(4,3)	B(4,4)	B(4,5)	B(4,6)	A(7)	B(4,8)	B(4,9)	B(4,10)	B(4,11)	B(4,12)	A(13)	B(4,14)	B(4,15)
Conn_cashier_customer	A(1)	B(5,2)	B(5,3)	B(5,4)	B(5,5)	B(5,6)	B(5,7)	B(5,8)	B(5,9)	B(5,10)	B(5,11)	B(5,12)	A(13)	B(5,14)	B(5,15)

STEP 4. FDcashier:{cashier_id} --->{cashier_name,cashier_address,cashier_bancaccount,cashier_insurance,cashier_salary}

FDcashier:{cashier_id} --->{cashier_name,cashier_address,cashier_bancaccount,cashier_insurance,cashier_salary} STEP4														
		cashier_id	cashier_name	cashier_address	cashier_bankaccount	cashier_insurance	cashier_salary	movie_id	movie_name	movie_language	movie_duration	movie_type	movie_date	
Cashier	A(1)	A(2)	A(3)	A(4)	A(5)	A(6)	B(1,7)	B(1,8)	B(1,9)	B(1,10)	B(1,11)	B(1,12)	B(1,13)	B
Movie	B(2,1)	B(2,2)	B(2,3)	B(2,4)	B(2,5)	B(2,6)	A(7)	A(8)	A(9)	A(10)	A(11)	A(12)	B(2,13)	B
Customer	B(3,1)	B(3,2)	B(3,3)	B(3,4)	B(3,5)	B(3,6)	B(3,7)	B(3,8)	B(3,9)	B(3,10)	B(3,11)	B(3,12)	A(13)	A
Booking ticket	A(1)	A(2)	A(3)	A(4)	A(5)	A(6)	A(7)	B(4,8)	B(4,9)	B(4,10)	B(4,11)	B(4,12)	A(13)	B
Conn_cashier_customer	A(1)	A(2)	A(3)	A(4)	A(5)	A(6)	B(5,7)	B(5,8)	B(5,9)	B(5,10)	B(5,11)	B(5,12)	A(13)	B

STEP 5. FD_{movie}:{movie_id} --->{movie_name,cashier_address,cashier_bancaccount,cashier_insurance,cashier_salary}

FD_{movie}:{movie_id} --->{movie_name,cashier_address,cashier_bancaccount,cashier_insurance,cashier_salary} STEP 5												
Cashier	A(1)	A(2)	A(3)	A(4)	A(5)	A(6)	B(1,7)	B(1,8)	B(1,9)	B(1,10)	B(1,11)	B(1,12)
Movie	B(2,1)	B(2,2)	B(2,3)	B(2,4)	B(2,5)	B(2,6)	A(7)	A(8)	A(9)	A(10)	A(11)	A(12)
Customer	B(3,1)	B(3,2)	B(3,3)	B(3,4)	B(3,5)	B(3,6)	B(3,7)	B(3,8)	B(3,9)	B(3,10)	B(3,11)	B(3,12)
Booking ticket	A(1)	A(2)	A(3)	A(4)	A(5)	A(6)	A(7)	A(8)	A(9)	A(10)	A(11)	A(12)
Conn_cashier_customer	A(1)	A(2)	A(3)	A(4)	A(5)	A(6)	B(5,7)	B(5,8)	B(5,9)	B(5,10)	B(5,11)	B(5,12)

STEP 6. $FD_{customer}$:{customer_id} --->{customer_name,customer_mobile,customer_email,customer_address,customer_discountTicketPrice}

Cashier	B(1,13)	B(1,14)	B(1,15)	B(1,16)	B(1,17)	B(1,18)
Movie	B(2,13)	B(2,14)	B(2,15)	B(2,16)	B(2,17)	B(2,18)
Customer	A(13)	A(14)	A(15)	A(16)	A(17)	A(18)
Booking ticket	A(13)	A(14)	A(15)	A(16)	A(17)	A(18)
Conn_cashier_customer	A(13)	A(14)	A(15)	A(16)	A(17)	A(18)

STEP 7. FD_{ticket}:{ticket_id} --->{ticket_dateTime,ticket_venue,ticket_number,customer_ticket(**FK** - ref : customer_id),cashier_ticket(**FK** - ref : cashier_id),movie_ticket(**FK** - ref : movie_id)}

Cashier	B(1,19)	B(1,20)	B(1,21)	B(1,22)	B(1,23)	B(1,24)	B(1,25)
Movie	B(2,19)	B(2,20)	B(2,21)	B(2,22)	B(2,23)	B(2,24)	B(2,25)
Customer	B(3,19)	B(3,20)	B(3,21)	B(3,22)	B(3,23)	B(3,24)	B(3,25)
Booking ticket	A(19)	A(20)	A(21)	A(22)	A(23)	A(24)	A(25)
Conn_cashier_customer	B(5,19)	B(5,20)	B(5,21)	B(5,22)	B(5,23)	B(5,24)	B(5,25)

STEP 8. $Fd_{conn\_cashier\_customer}$:{conn_id} --->{conn_cashier(ref : cashier_id),conn_customer(ref : customer_id)}

Cashier	B(1,26)	B(1,27)	B(1,28)
Movie	B(2,26)	B(2,27)	B(2,28)
Customer	B(3,26)	B(3,27)	B(3,28)
Booking ticket	A(26)	A(27)	A(28)
Conn_cashier_customer	A(26)	A(27)	A(28)

THE END

	cashier_id	cashier_name	cashier_address	cashier_bankaccount	cashier_insurance	cashier_salary	movie_id	movie_name	movie_language	movie_duration	movie_type	movie_date	customer_id	customer_name	customer_mobile	customer_email	customer_address	customer_discountTicketPrice	ticket_id	
THE END																				
Cashier	A(1)	A(2)	A(3)	A(4)	A(5)	A(6)	B(1,7)	B(1,8)	B(1,9)	B(1,10)	B(1,11)	B(1,12)	B(1,13)	B(1,14)	B(1,15)	B(1,16)	B(1,17)	B(1,18)	B(1,19)	B
Movie	B(2,1)	B(2,2)	B(2,3)	B(2,4)	B(2,5)	B(2,6)	A(7)	A(8)	A(9)	A(10)	A(11)	A(12)	B(2,13)	B(2,14)	B(2,15)	B(2,16)	B(2,17)	B(2,18)	B(2,19)	B
Customer	B(3,1)	B(3,2)	B(3,3)	B(3,4)	B(3,5)	B(3,6)	B(3,7)	B(3,8)	B(3,9)	B(3,10)	B(3,11)	B(3,12)	A(13)	A(14)	A(15)	A(16)	A(17)	A(18)	B(3,19)	B
Booking ticket	A(1)	A(2)	A(3)	A(4)	A(5)	A(6)	A(7)	A(8)	A(9)	A(10)	A(11)	A(12)	A(13)	A(14)	A(15)	A(16)	A(17)	A(18)	A(19)	A
Conn_cashier_customer	A(1)	A(2)	A(3)	A(4)	A(5)	A(6)	B(5,7)	B(5,8)	B(5,9)	B(5,10)	B(5,11)	B(5,12)	A(13)	A(14)	A(15)	A(16)	A(17)	A(18)	B(5,19)	B

So, my project is lossless decomposition.

DATABASE

Creating the table

```
CREATE TABLE cashier(
employee_id int primary key,
employee_name varchar(100),
employee_address varchar(100),
employee_bankaccount varchar(100),
employee_insurance varchar(100),
employee_salary int
);

CREATE TABLE movie(
movie_id int primary key,
movie_name varchar(100),
movie_language varchar(50),
movie_duration int,
movie_type varchar(100),
movie_date date
);

CREATE TABLE customer(
customer_id int primary key,
customer_name varchar(100),
customer_mobile varchar(30),
customer_email varchar(100),
customer_address varchar(100),
customer_discountPrice int
);

CREATE TABLE booking_ticket(
ticket_id int primary key,
ticket_date datetime,          --correct it
ticket_venue int,
ticket_number int,
customer_ticket int NOT NULL references customer(customer_id),
cashier_ticket int NOT NULL references cashier(employee_id),
movie_ticket int NOT NULL references movie(movie_id)
);

CREATE TABLE conn_cashier_customer(
conn_id int identity primary key,
conn_cashier int references cashier(employee_id),
conn_customer int references customer(customer_id),
CONSTRAINT  uq_conn_cashier_customer UNIQUE(conn_cashier,conn_customer)
);
```

Query 1: List cashiers and their data

```
SELECT * FROM cashier ;
```

	employee_id	employee_name	employee_address	employee_bankaccount	employee_insurance	employee_salary
1	1	Ally Smith	Csengery 82	5443213-5642219	A65B7	100000
2	2	Meg WilL	Becsi 44	5643457-9876515	B98C6	10500
3	3	Alan Johnson	Bertalan 67	1238765-1753457	N76M9	110000
4	4	Alex Pitt	Will 54	1238765-1753459	H76M0	150000
5	5	Bekbolsun Botoev	Sputni 16	1238765-2753457	M36M9	200000

Query 2: List customers and their data

```
SELECT * FROM customer;
```

	customer_id	customer_name	customer_mobile	customer_email	customer_address	customer_discountPrice
1	1	Kate Wilson	706102030	kate@gmail.com	Istvan 10	1000
2	2	Anne Hath	708105030	anne@gmail.com	Deak 15	4000
3	3	Anton Pavel	709105010	anton@gmail.c...	Dorothy 18	3000
4	4	Alymkan Boto...	709809023	alymkan@gmail...	Sputnik 17	5000
5	5	Aruuke Alimb...	719105010	aruuke@gmail....	Koibagarov 19	1000

Query 3: List movies and their data

```
SELECT * FROM movie;
```

	movie_id	movie_name	movie_language	movie_duration	movie_type	movie_date
1	1	Twilight	English	90	fantasy	2020-10-18
2	2	Hidden figures	German	100	drama	2020-10-14
3	3	Peculiar chil...	Russian	120	fantasy	2020-10-19
4	4	Darak yry	Kyrgyz	120	historical ...	2020-10-20
5	5	Kurmanjan ...	Kyrgyz	120	historical ...	2020-10-21

Query 4: List customer names with their average price of tickets

```
SELECT customer_name, sum(customer_discountPrice) as summ
FROM customer
GROUP BY customer_name;
```

	customer_name	summ
1	Alymkan Botoeva	4000
2	Anne Hath	5000
3	Anton Pavel	3000
4	Kate Wilson	1000

Query 5: Using row level functions,

```
SELECT movie_name,movie_date,
concat(datediff(month,movie_date,getdate()),'month(s)') as passed
FROM movie
```

	movie_name	movie_date	passed
1	Twilight	2020-10-18	1month(s)
2	Hidden figures	2020-10-14	1month(s)
3	Peculiar chil...	2019-10-19	13mont...
4	Darak yry	2019-10-20	13mont...
5	Kurmanjan D...	2021-10-21	-11mont...
6	Cinderella	2019-10-21	13mont...

Query 6: Count movies by their movie language

```
SELECT count(movie_name) as numberOfFilms,movie_language
FROM movie
GROUP BY movie_language;
```

	numberOfFilms	movie_language
1	1	English
2	1	German
3	2	Kyrgyz
4	1	Russian

Query 7: Count movies by their year

```
SELECT count(1) as numOfmovie,year(movie_date) as years
FROM movie
GROUP BY datepart(year,movie_date);
```

	numOfmovie	years
1	2	2019
2	2	2020
3	1	2021

Query 8: Find the venue of customer whose name is Alymkan Botoeva

```
SELECT ticket_venue as venue
FROM customer INNER JOIN booking_ticket ON(customer_id=customer_ticket)
WHERE customer_name='Alymkan Botoeva';
```

	venue
1	13

Query 9: List customer name and movie name where movie is Kyrgyz language

```
SELECT customer_name,movie_name
 FROM booking_ticket INNER JOIN customer ON(customer_id=customer_ticket)
                     INNER JOIN movie ON(movie_id=movie_ticket)
 WHERE movie_language like 'Kyrgyz';
```

	customer_name	movie_name
1	Anne Hath	Darak yry
2	Alymkan Botoeva	Kurmanjan Datka
3	Anne Hath	Kurmanjan Datka
4	Kate Wilson	Kurmanjan Datka

Query 10: List customer name and cashier name where ticket price is 1000 Forint

```
SELECT LOWER(customer_name),employee_name
 FROM customer INNER JOIN conn_cashier_customer ON(customer_id=conn_customer)
               INNER JOIN cashier ON(employee_id=conn_cashier)
 WHERE customer_discountPrice=1000;
```

	(No column name)	employee_name
1	kate wilson	Ally Smith
2	anne hath	Ally Smith
3	kate wilson	Meg WilL
4	anne hath	Alan Johnson

QUERY 11: List the customers who chose the fantasy movie and date time and list the cashiers who served

```
SELECT distinct upper(customer_name),employee_name,movie_name, ticket_date
FROM conn_cashier_customer INNER JOIN customer ON(conn_customer=customer_id)
                           INNER JOIN cashier ON(conn_cashier=employee_id),
                     movie  INNER JOIN booking_ticket ON(movie_id=movie_ticket)
WHERE movie_type='fantasy';
```

	customername	employee_name	movie_name	ticket_date
1	ANNE HATH	Alan Johnson	Peculiar children	2020-10-19 17:30:00.000
2	ANNE HATH	Alan Johnson	Twilight	2020-10-14 11:00:00.000
3	ANNE HATH	Ally Smith	Peculiar children	2020-10-19 17:30:00.000
4	ANNE HATH	Ally Smith	Twilight	2020-10-14 11:00:00.000
5	ANNE HATH	Meg WilL	Peculiar children	2020-10-19 17:30:00.000
6	ANNE HATH	Meg WilL	Twilight	2020-10-14 11:00:00.000
7	ANTON PAV...	Alex Pitt	Peculiar children	2020-10-19 17:30:00.000
8	ANTON PAV...	Alex Pitt	Twilight	2020-10-14 11:00:00.000

Query 12: List customer name and cashier name where cashiers' salary is 100000

```
SELECT customer_name, employee_name
FROM conn_cashier_customer INNER JOIN customer ON(conn_customer=customer_id)
                           INNER JOIN cashier ON(conn_cashier=employee_id)
WHERE employee_salary=100000;
```

	customer_name	employee_name
1	Kate Wilson	Ally Smith
2	Anne Hath	Ally Smith
3	Anton Pavel	Ally Smith

Query 13: List the cashiers whose salary is bigger than average

```
SELECT employee_name, employee_salary
FROM cashier
WHERE employee_salary >(SELECT avg(employee_salary) FROM cashier);
```

	employee_name	employee_salary
1	Alan Johnson	110000
2	Bekbolsun Botoev	200000

Query 14: Maximum number of customers who chose the fantasy movie

```
SELECT max(calculate.numberOfPeople) as maximum
FROM (SELECT customer_name,  count(1) as numberOfPeople,customer_id
      FROM conn_cashier_customer INNER JOIN customer ON(conn_customer=customer_id)
                                 INNER JOIN cashier ON(conn_cashier=employee_id),
                           movie  INNER JOIN booking_ticket ON(movie_id=movie_ticket)
      WHERE movie_type='fantasy'
      GROUP BY customer_name,customer_id  ) as calculate
                                 INNER JOIN customer ON(calculate.customer_id=customer.customer_id);
```

	maximum
1	8

Query 15: Minimum number of customers who chose the fantasy movie

```
SELECT min(calculate.numberOfPeople) as minimum
FROM (SELECT customer_name,  count(1) as numberOfPeople,customer_id
      FROM conn_cashier_customer INNER JOIN customer ON(conn_customer=customer_id)
                                 INNER JOIN cashier ON(conn_cashier=employee_id),
                           movie  INNER JOIN booking_ticket ON(movie_id=movie_ticket)
      WHERE movie_type='fantasy'
      GROUP BY customer_name,customer_id  ) as calculate
                                 INNER JOIN customer ON(calculate.customer_id=customer.customer_id);
```

	minimum
1	4

Query 16: Calculate average that how many customers chose that movie language is Kyrgyz or Russian

```
SELECT avg(numberOfPeople) as average
FROM (SELECT customer_name,  count(1) as numberOfPeople,customer_id
      FROM conn_cashier_customer INNER JOIN customer ON(conn_customer=customer_id)
                                 INNER JOIN cashier ON(conn_cashier=employee_id),
                           movie  INNER JOIN booking_ticket ON(movie_id=movie_ticket)
      WHERE movie_language IN ('Kyrgyz','Russian')
      GROUP BY customer_name,customer_id  ) as calculate
                                 INNER JOIN customer ON(calculate.customer_id=customer.customer_id);
```

	average
1	16

Query 17: Number of people who chose the film in the 2020-10-18

```
SELECT calculate.numberOfPeople as number
FROM (SELECT customer_name,  count(customer_id) as numberOfPeople,customer_id
      FROM conn_cashier_customer INNER JOIN customer ON(conn_customer=customer_id)
                                 INNER JOIN cashier ON(conn_cashier=employee_id),
                           movie  INNER JOIN booking_ticket ON(movie_id=movie_ticket)
      WHERE movie_date='2020-10-18'
      GROUP BY customer_name, customer_id  ) as calculate
                                 INNER JOIN customer ON(calculate.customer_id=customer.customer_id);
```

	number
1	2
2	3
3	4
4	2

Query 18:Calculate the number of movies that a customer watched

```
--
SELECT customer_name, grouping(customer_name) as nameGrouping,
count(customer_name) as numberOfWatching,movie_name, grouping(movie_name) as movieGrouping
      FROM conn_cashier_customer INNER JOIN customer ON(conn_customer=customer_id)
                                 INNER JOIN cashier ON(conn_cashier=employee_id),
                           movie  INNER JOIN booking_ticket ON(movie_id=movie_ticket)
      WHERE movie_type='fantasy'
      GROUP BY GROUPING SETS((customer_name,movie_name),(customer_name)) ;
```

	customer_name	nameGrouping	numberOfWatching	movie_name	movieGrouping
1	Anne Hath	0	5	Peculiar children	0
2	Anne Hath	0	5	Twilight	0
3	Anne Hath	0	10	NULL	1
4	Anton Pavel	0	4	Peculiar children	0
5	Anton Pavel	0	4	Twilight	0
6	Anton Pavel	0	8	NULL	1
7	Kate Wilson	0	2	Peculiar children	0
8	Kate Wilson	0	2	Twilight	0

Query 19: Listing number of watching of a customer the same movie, number of watching of each movie, number of watching of fantasy movies, number of watching of each person that watched fantasy movie

```
SELECT customer_name, grouping(customer_name) as nameGrouping,
count(customer_name) as numberOfWatching,movie_name, grouping(movie_name) as movieGrouping
      FROM conn_cashier_customer INNER JOIN customer ON(conn_customer=customer_id)
                                 INNER JOIN cashier ON(conn_cashier=employee_id),
                          movie  INNER JOIN booking_ticket ON(movie_id=movie_ticket)
      WHERE movie_type='fantasy'
      GROUP BY CUBE(customer_name,movie_name);
```

	customer_name	nameGrouping	numberOfWatching	movie_name	movieGrouping
1	Anne Hath	0	5	Peculiar children	0
2	Anton Pavel	0	4	Peculiar children	0
3	Kate Wilson	0	2	Peculiar children	0
4	NULL	1	11	Peculiar children	0
5	Anne Hath	0	5	Twilight	0
6	Anton Pavel	0	4	Twilight	0
7	Kate Wilson	0	2	Twilight	0
8	NULL	1	11	Twilight	0
9	NULL	1	22	NULL	1
10	Anne Hath	0	10	NULL	1
11	Anton Pavel	0	8	NULL	1
12	Kate Wilson	0	4	NULL	1

Query 20: : Listing number of watching of a customer the same movie, number of watching of each person that watched fantasy movie

```
SELECT customer_name, grouping(customer_name) as nameGrouping,
count(customer_name) as numberOfWatching,movie_name, grouping(movie_name) as movieGrouping
      FROM conn_cashier_customer INNER JOIN customer ON(conn_customer=customer_id)
                                 INNER JOIN cashier ON(conn_cashier=employee_id),
                           movie  INNER JOIN booking_ticket ON(movie_id=movie_ticket)
      WHERE movie_type='fantasy'
      GROUP BY ROLLUP(customer_name,movie_name);
```

	customer_name	nameGrouping	numberOfWatching	movie_name	movieGrouping
1	Anne Hath	0	5	Peculiar children	0
2	Anne Hath	0	5	Twilight	0
3	Anne Hath	0	10	NULL	1
4	Anton Pavel	0	4	Peculiar children	0
5	Anton Pavel	0	4	Twilight	0
6	Anton Pavel	0	8	NULL	1
7	Kate Wilson	0	2	Peculiar children	0
8	Kate Wilson	0	2	Twilight	0
9	Kate Wilson	0	4	NULL	1
10	NULL	1	22	NULL	1

Query 21: Listing number of watching of a customer the same movie, number of watching of each movie, number of watching of fantasy movies

```
SELECT customer_name, grouping(customer_name) as nameGrouping,
count(customer_name) as numberOfWatching,movie_name, grouping(movie_name) as movieGrouping
      FROM conn_cashier_customer INNER JOIN customer ON(conn_customer=customer_id)
                                 INNER JOIN cashier ON(conn_cashier=employee_id),
                           movie  INNER JOIN booking_ticket ON(movie_id=movie_ticket)
      WHERE movie_type='fantasy'
      GROUP BY ROLLUP(movie_name,customer_name);
```

	customer_name	nameGrouping	numberOfWatching	movie_name	movieGrouping
1	Anne Hath	0	5	Peculiar children	0
2	Anton Pavel	0	4	Peculiar children	0
3	Kate Wilson	0	2	Peculiar children	0
4	NULL	1	11	Peculiar children	0
5	Anne Hath	0	5	Twilight	0
6	Anton Pavel	0	4	Twilight	0
7	Kate Wilson	0	2	Twilight	0
8	NULL	1	11	Twilight	0
9	NULL	1	22	NULL	1

Query 22: List the number of movie by language, name, type

```
SELECT count(movie_name) as numberOfFilms,movie_language, GROUPING(movie_language)as languageGrouping,
movie_name, GROUPING(movie_name) as nameGrouping,movie_type, GROUPING(movie_type) as typeGrouping
FROM movie
GROUP BY GROUPING SETS (( movie_language,movie_name,movie_type),(movie_language));
```

	numberOfFilms	movie_language	languageGrouping	movie_name	nameGrouping	movie_type	typeGrouping
1	1	English	0	Cinderella	0	cartoon	0
2	1	English	0	Twilight	0	fantasy	0
3	2	English	0	NULL	1	NULL	1
4	1	German	0	Hidden fig...	0	drama	0
5	1	German	0	NULL	1	NULL	1
6	1	Kyrgyz	0	Darak yry	0	historical ...	0
7	1	Kyrgyz	0	Kurmanjan...	0	historical ...	0
8	2	Kyrgyz	0	NULL	1	NULL	1
9	1	Russian	0	Peculiar c...	0	fantasy	0
10	1	Russian	0	NULL	1	NULL	1

Query 23

UPDATE 1

```
UPDATE customer
SET customer_discountPrice= round(((4000+100-1)*rand()+100),0)
WHERE customer_id=2;
SELECT* FROM customer;
```

Before the update

	customer_id	customer_name	customer_mobile	customer_email	customer_address	customer_discountPrice
1	1	Kate Wilson	706102030	kate@gmail.com	Istvan 10	1000
2	2	Anne Hath	708105030	anne@gmail.com	Deak 15	4000
3	3	Anton Pavel	709105010	anton@gmail.c...	Dorothy 18	3000
4	4	Alymkan Boto...	709809023	alymkan@gmail...	Sputnik 17	5000
5	5	Aruuke Alimb...	719105010	aruuke@gmail....	Koibagarov 19	1000

After the update

	customer_id	customer_name	customer_mobile	customer_email	customer_address	customer_discountPri
1	1	Kate Wilson	706102030	kate@gmail.com	Istvan 10	1000
2	2	Anne Hath	708105030	anne@gmail.c...	Deak 15	2101
3	3	Anton Pavel	709105010	anton@gmail.c...	Dorothy 18	3000
4	4	Alymkan Boto...	709809023	alymkan@gma...	Sputnik 17	4000
5	5	Anne Hath	708105030	anne@gmail.c...	Deak 15	1000

Query 24

Update 2. Update cashiers' salary that customers who watch "Kurmanjan Datka"

```
SET employee_salary=employee_salary*2
WHERE employee_id IN
(
SELECT cashier.employee_id
FROM (
       SELECT employee_id, count(customer_email) as numberOfEmail
       FROM cashier INNER JOIN conn_cashier_customer ON(conn_cashier=employee_id)
                    INNER JOIN customer ON(conn_customer=customer_id)
       WHERE customer_email='alymkan@gmail.com' or customer_email='anton@gmail.com'
       GROUP BY employee_id
     )as clients
     INNER JOIN cashier ON(cashier.employee_id=clients.employee_id),
     movie  INNER JOIN booking_ticket ON(movie_id=movie_ticket)
     WHERE movie_name='Kurmanjan Datka'
)
SELECT *FROM cashier;
```

Before update

	employee_id	employee_name	employee_address	employee_bankaccount	employee_insurance	employee_salary
1	1	Ally Smith	Csengery 82	5443213-5642219	A65B7	100000
2	2	Meg WilL	Becsi 44	5643457-9876515	B98C6	10500
3	3	Alan Johnson	Bertalan 67	1238765-1753457	N76M9	110000
4	4	Alex Pitt	Will 54	1238765-1753459	H76M0	150000
5	5	Bekbolsun Botoev	Sputni 16	1238765-2753457	M36M9	200000

After update

	employee_id	employee_name	employee_address	employee_bankaccount	employee_insurance	employee_salary
1	1	Ally Smith	Csengery 82	5443213-5642219	A65B7	200000
2	2	Meg WilL	Becsi 44	5643457-9876515	B98C6	21000
3	3	Alan Johnson	Bertalan 67	1238765-1753457	N76M9	110000
4	4	Alex Pitt	Will 54	1238765-1753459	H76M0	200000
5	5	Bekbolsun Bot...	Sputni 16	1238765-2753457	M36M9	400000

Query 25

Insert 1: Insert new row to the movie table

```
INSERT INTO movie VALUES(6,'Cinderella','English', 90,'cartoon','2019-10-21');
SELECT * FROM movie;
```

	movie_id	movie_name	movie_language	movie_duration	movie_type	movie_date
1	1	Twilight	English	90	fantasy	2020-10-18
2	2	Hidden fig...	German	100	drama	2020-10-14
3	3	Peculiar c...	Russian	120	fantasy	2019-10-19
4	4	Darak yry	Kyrgyz	120	historical ...	2019-10-20
5	5	Kurmanjan...	Kyrgyz	120	historical ...	2021-10-21
6	6	Cinderella	English	90	cartoon	2019-10-21

Insert 2: Insert new record for customer

```
INSERT INTO customer(customer_id,customer_name,customer_discountPrice)
VALUES(6,'Begai Manas kyzy',
(SELECT distinct customer_discountPrice*2
FROM customer
WHERE customer_discountPrice=1000
));
SELECT * FROM customer;
```

	customer_id	customer_name	customer_mobile	customer_email	customer_address	customer_discountPrice
1	1	Kate Wilson	706102030	kate@gmail.com	Istvan 10	1000
2	2	Anne Hath	708105030	anne@gmail.c...	Deak 15	808
3	3	Anton Pavel	709105010	anton@gmail.c...	Dorothy 18	3000
4	4	Alymkan Boto...	709809023	alymkan@gma...	Sputnik 17	4000
5	5	Anne Hath	708105030	anne@gmail.c...	Deak 15	1000
6	6	Begai Manas ...	NULL	NULL	NULL	2000

Query 26

Delete 1:

```
DELETE booking_ticket from booking_ticket WHERE ticket_id=9;
SELECT *FROM booking_ticket;
```

Before deletion

	ticket_id	ticket_date	ticket_venue	ticket_number	customer_ticket	cashier_ticket	movie_ticket
2	2	2020-10-14 11:00:00.000	9	8	3	2	1
3	3	2020-10-19 17:30:00.000	11	2	2	1	3
4	4	2020-10-19 12:10:00.000	18	9	5	4	4
5	5	2020-10-19 14:50:00.000	13	10	4	5	5
6	6	2020-10-19 17:30:00.000	18	7	4	2	3
7	7	2020-10-19 17:30:00.000	15	6	5	5	5
8	8	2020-10-19 17:30:00.000	19	12	3	5	2
9	9	2020-10-19 17:30:00.000	10	2	1	4	5

After deletion

	ticket_id	ticket_date	ticket_venue	ticket_number	customer_ticket	cashier_ticket	movie_ticket
1	1	2020-10-18 15:20:00.000	5	7	1	3	2
2	2	2020-10-14 11:00:00.000	9	8	3	2	1
3	3	2020-10-19 17:30:00.000	11	2	2	1	3
4	4	2020-10-19 12:10:00.000	18	9	5	4	4
5	5	2020-10-19 14:50:00.000	13	10	4	5	5
6	6	2020-10-19 17:30:00.000	18	7	4	2	3
7	7	2020-10-19 17:30:00.000	15	6	5	5	5
8	8	2020-10-19 17:30:00.000	19	12	3	5	2

Query 27

Delete 2: Delete ticketId where ticket price is 1000 Forint

```
SELECT*FROM booking_ticket;
DELETE booking_ticket FROM booking_ticket
            INNER JOIN customer ON(customer_id=customer_ticket)
WHERE customer_id IN
(
SELECT ticket_id
FROM booking_ticket INNER JOIN customer ON(customer_id=customer_ticket)
WHERE customer_discountPrice=1000
)
SELECT*FROM booking_ticket;
```

Before deletion

	ticket_id	ticket_date	ticket_venue	ticket_number	customer_ticket	cashier_ticket	movie_ticket
1	1	2020-10-18 15:20:00.000	5	7	1	3	2
2	2	2020-10-14 11:00:00.000	9	8	3	2	1
3	3	2020-10-19 17:30:00.000	11	2	2	1	3
4	4	2020-10-19 12:10:00.000	18	9	5	4	4
5	5	2020-10-19 14:50:00.000	13	10	4	5	5
6	6	2020-10-19 17:30:00.000	18	7	4	2	3
7	7	2020-10-19 17:30:00.000	15	6	5	5	5
8	8	2020-10-19 17:30:00.000	19	12	3	5	2

After deletion

DATABASE

	ticket_id	ticket_date	ticket_venue	ticket_number	customer_ticket	cashier_ticket	movie_ticket
1	2	2020-10-14 11:00:00.000	9	8	3	2	1
2	3	2020-10-19 17:30:00.000	11	2	2	1	3
3	4	2020-10-19 12:10:00.000	18	9	5	4	4
4	7	2020-10-19 17:30:00.000	15	6	5	5	5
5	8	2020-10-19 17:30:00.000	19	12	3	5	2

SUMMARY

SIMPLE SINGLE-TABLE SELECT: QUERY #1 -QUERY 3

SIMPLE SINGLE-TABLE GROUP BY: QUERY #4 -QUERY #8

COMPLEX MULTI-TABLE SELECT: QUERY #9 – QUERY 12

COMPLEX SUBQUARY SELECT: QUERY #13 – QUERY #17

ANALYTICS/ADVANCED GROUPING: QUERY #18 – QUERY #22

DML OPERATIONS:

UPDATE: QUERY #23 – QUERY #24

INSERT: QUERY #25 – QUERY #26

DELETE: QUERY #27 – QUERY #28

USED AGGREGATE FUNCTIONS:

- AVG(expression) Calculate the average of the expression.
- COUNT(expression) Count occurrences of non-null values returned by the expression.
- COUNT(*) Counts all rows in the specified table.
- MIN(expression) Finds the minimum expression value.
- MAX(expression) Finds the maximum expression value.
- SUM(expression) Calculate the sum of the expression.

USED ROW LEVEL FUNCTIONS:

YEAR()	LOWER()
LIKE	UPPER
GETDATE()	ROUND()
DATEDIFF()	RAND()
DATEPART()	CONCAT()