НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ «КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО» ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра системного програмування і спеціалізованих комп'ютерних систем

Лабораторна робота №1

з дисципліни «Бази даних і засоби управління»

Тема: «**здобуття вмінь програмування прикладних додатків баз даних PostgreSQL**»

Виконали: студенти III курсу

ФПМ групи КВ-81

Ядуха Б.В.

Викладач: ПетрашенкоА.В.

Київ 2020

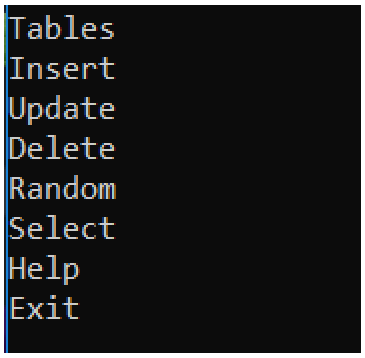
Метою роботи є здобуття вмінь програмування прикладних додатків баз даних PostgreSQL.

Загальне завдання роботи полягає у наступному:

1. Реалізувати функції внесення, редагування та вилучення даних у таблицях бази даних, створених у лабораторній роботі №1, засобами консольного інтерфейсу.
2. Передбачити автоматичне пакетне генерування «рандомізованих» даних у базі.
3. Забезпечити реалізацію пошуку за декількома атрибутами з двох та більше сутностей одночасно: для числових атрибутів – у рамках діапазону, для рядкових – як шаблон функції LIKE оператора SELECT SQL, для логічного типу – значення True/False, для дат – у рамках діапазону дат.
4. Програмний код виконати згідно шаблону MVC (модель-подання-контролер).

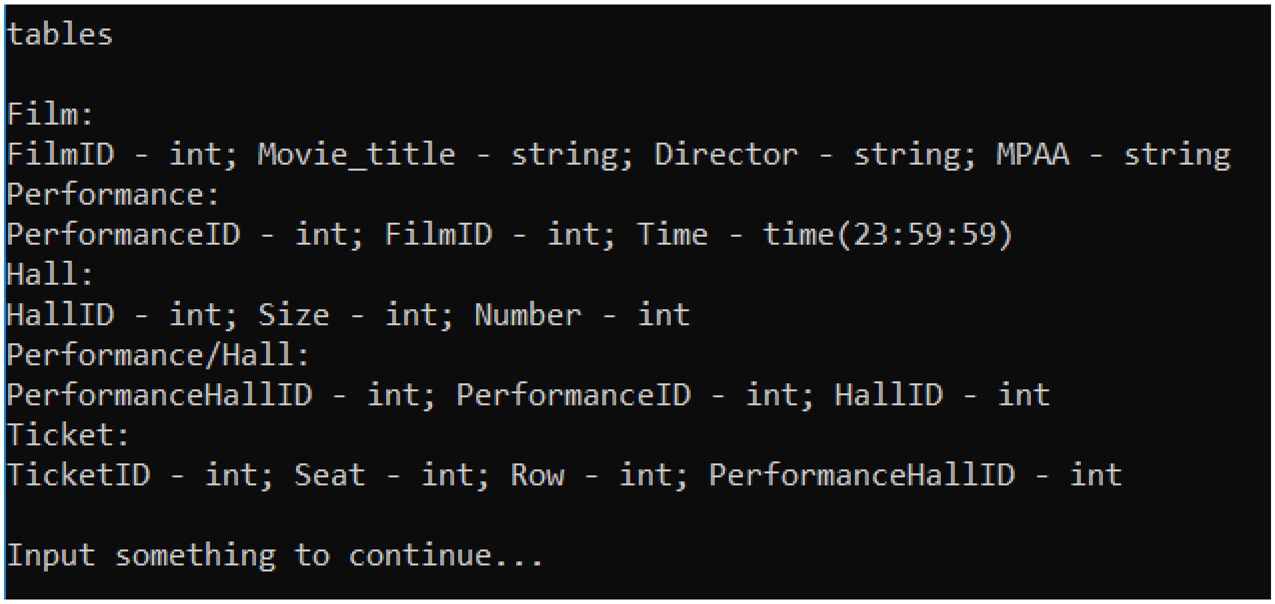
Завдання №1

Меню



```
Tables
Insert
Update
Delete
Random
Select
Help
Exit
```

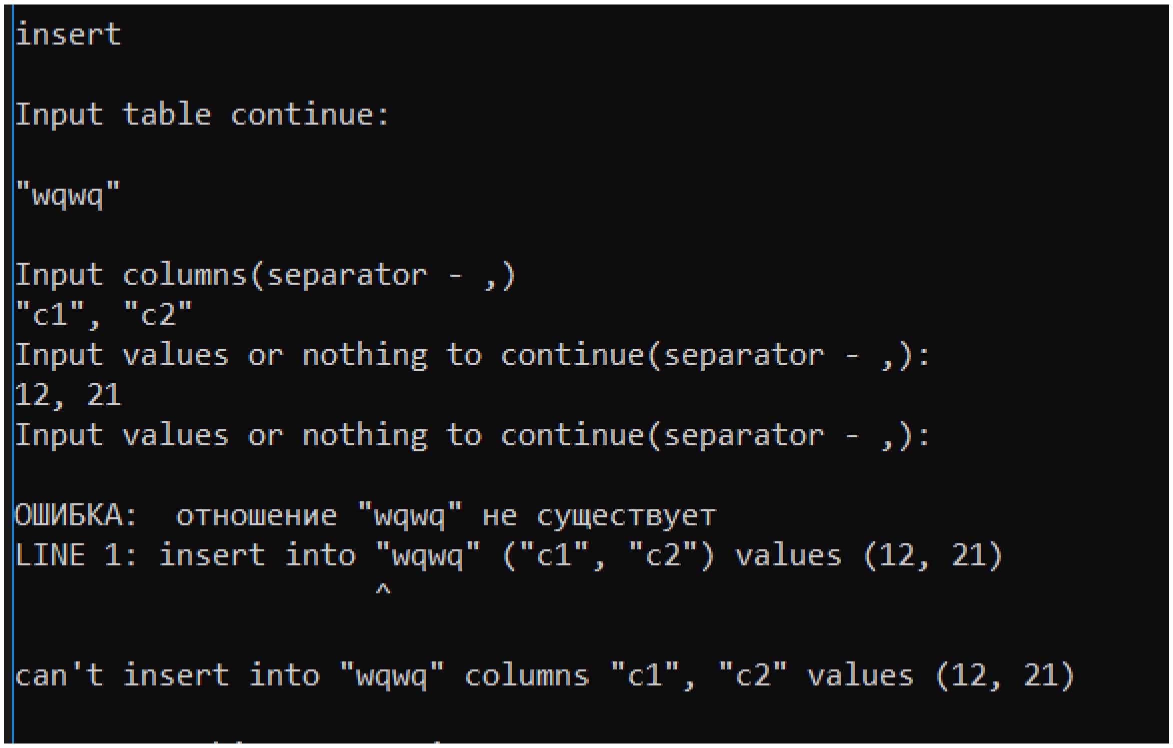
Інформація про таблиці:



```
tables

Film:
FilmID - int; Movie_title - string; Director - string; MPAA - string
Performance:
PerformanceID - int; FilmID - int; Time - time(23:59:59)
Hall:
HallID - int; Size - int; Number - int
Performance/Hall:
PerformanceHallID - int; PerformanceID - int; HallID - int
Ticket:
TicketID - int; Seat - int; Row - int; PerformanceHallID - int

Input something to continue...
```

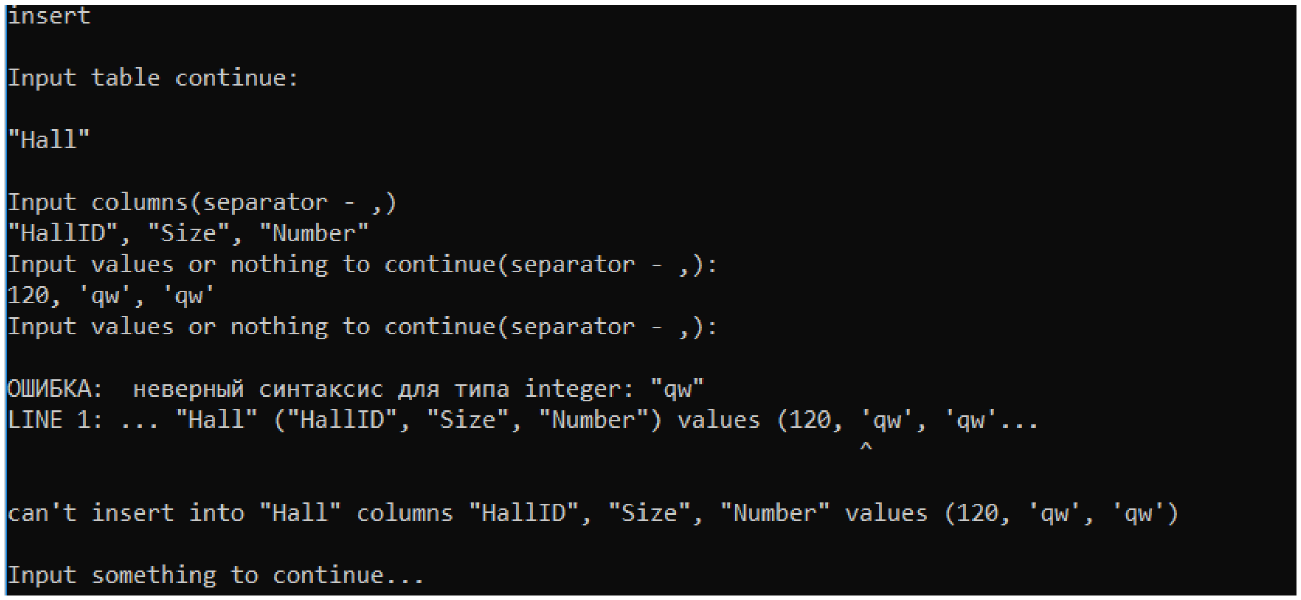
Insert:

При введенні неіснуючої таблиці



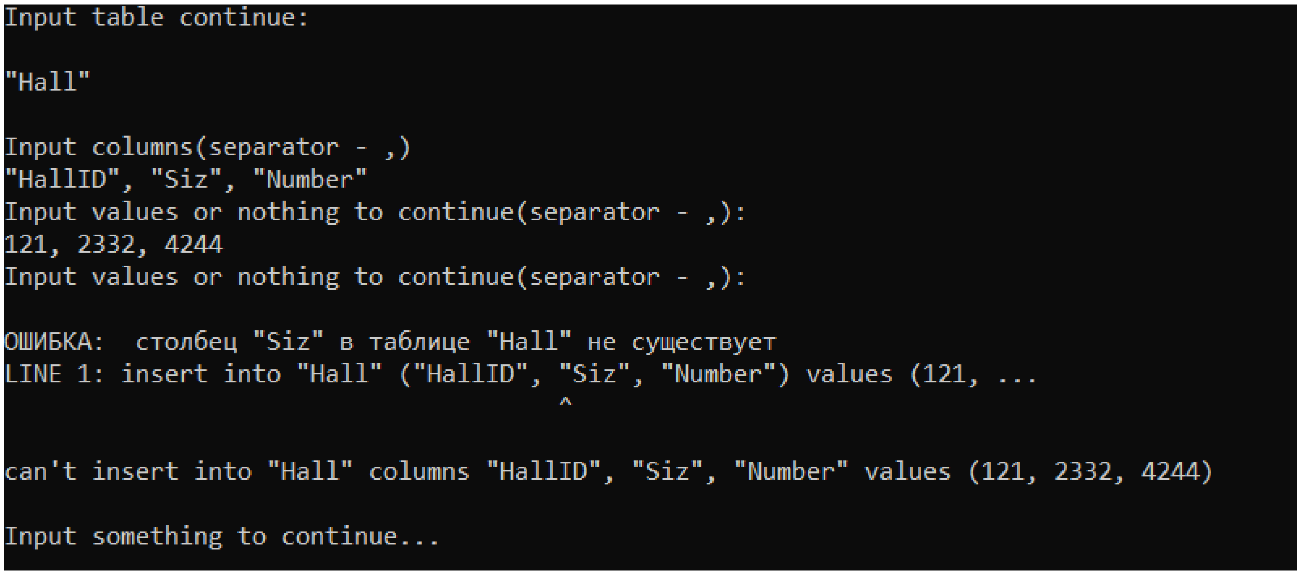
```
insert

Input table continue:

"wqwq"

Input columns(separator - ,)
"c1", "c2"
Input values or nothing to continue(separator - ,):
12, 21
Input values or nothing to continue(separator - ,):

ОШИБКА:  отношение "wqwq" не существует
LINE 1: insert into "wqwq" ("c1", "c2") values (12, 21)
                    ^

can't insert into "wqwq" columns "c1", "c2" values (12, 21)
```

При введенні невірного типу:



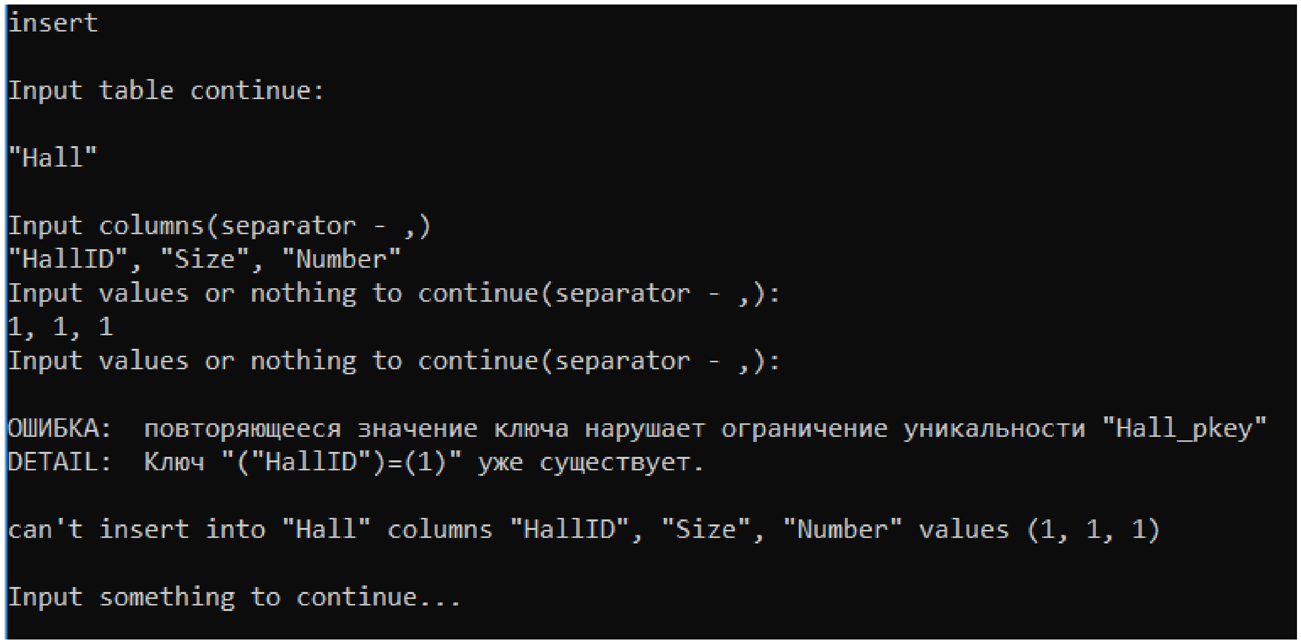
```
insert

Input table continue:

"Hall"

Input columns(separator - ,)
"HallID", "Size", "Number"
Input values or nothing to continue(separator - ,):
120, 'qw', 'qw'
Input values or nothing to continue(separator - ,):

ОШИБКА:  неверный синтаксис для типа integer: "qw"
LINE 1: ... "Hall" ("HallID", "Size", "Number") values (120, 'qw', 'qw'...
                                                             ^

can't insert into "Hall" columns "HallID", "Size", "Number" values (120, 'qw', 'qw')

Input something to continue...
```

При введенні неіснуючої колонки:



```
Input table continue:

"Hall"

Input columns(separator - ,)
"HallID", "Siz", "Number"
Input values or nothing to continue(separator - ,):
121, 2332, 4244
Input values or nothing to continue(separator - ,):

ОШИБКА:  столбец "Siz" в таблице "Hall" не существует
LINE 1: insert into "Hall" ("HallID", "Siz", "Number") values (121, ...
                                      ^

can't insert into "Hall" columns "HallID", "Siz", "Number" values (121, 2332, 4244)

Input something to continue...
```

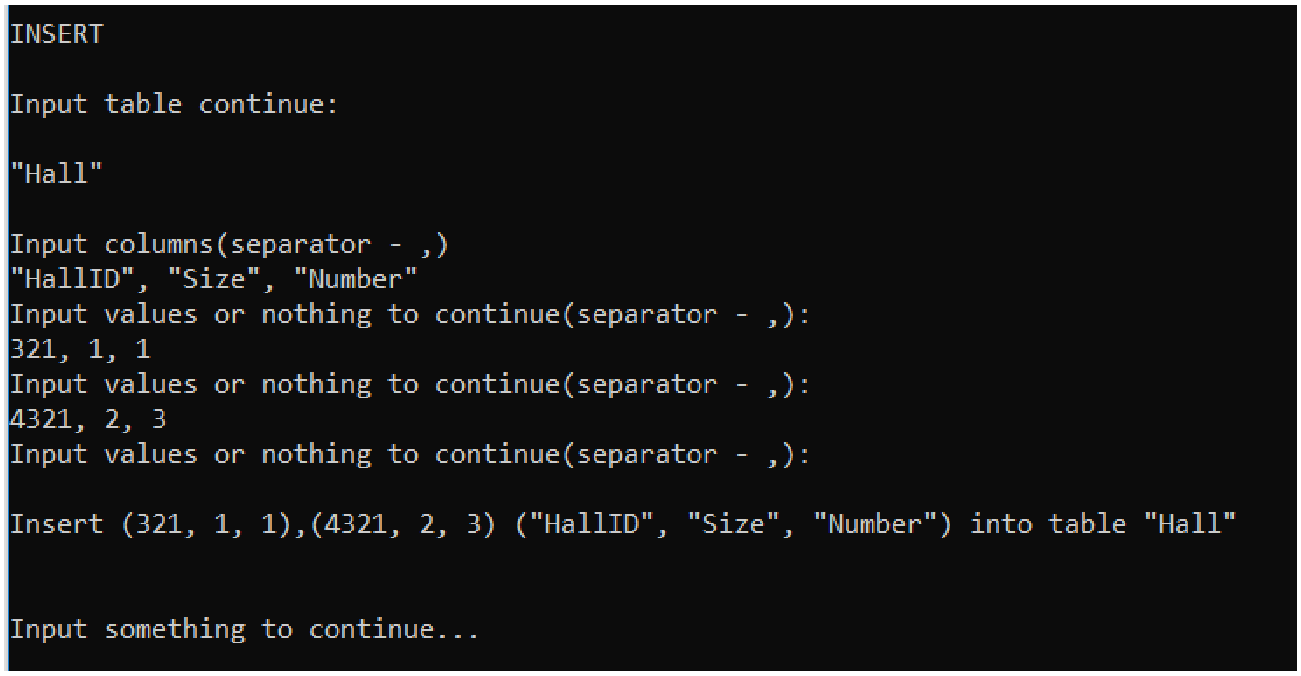
При введенні шснуючого первинного ключа:



```
insert

Input table continue:

"Hall"

Input columns(separator - ,)
"HallID", "Size", "Number"
Input values or nothing to continue(separator - ,):
1, 1, 1
Input values or nothing to continue(separator - ,):

ОШИБКА:  повторяющееся значение ключа нарушает ограничение уникальности "Hall_pkey"
DETAIL:  Ключ "("HallID")=(1)" уже существует.

can't insert into "Hall" columns "HallID", "Size", "Number" values (1, 1, 1)

Input something to continue...
```

Вивід помилок в інших запитах схожий, тому надалі будуть показані не усі помилкиі

При введенні коректного запиту:

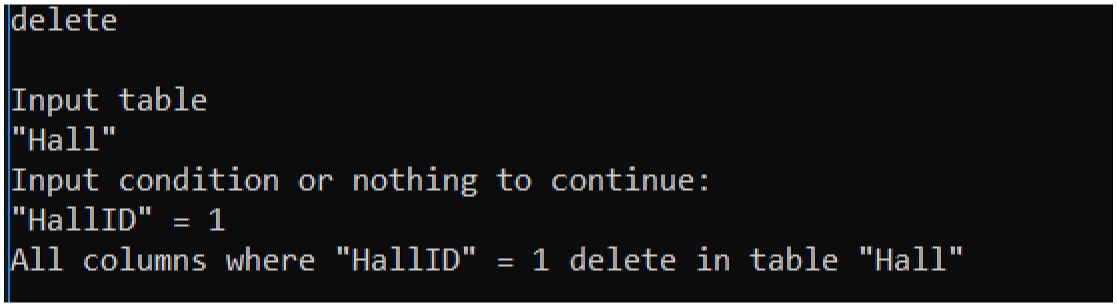


```
INSERT

Input table continue:

"Hall"

Input columns(separator - ,)
"HallID", "Size", "Number"
Input values or nothing to continue(separator - ,):
321, 1, 1
Input values or nothing to continue(separator - ,):
4321, 2, 3
Input values or nothing to continue(separator - ,):

Insert (321, 1, 1),(4321, 2, 3) ("HallID", "Size", "Number") into table "Hall"


Input something to continue...
```

Delete:

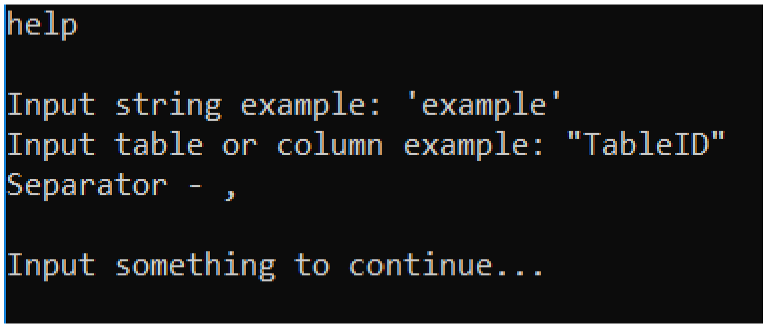
При видаленні батьківської таблиці видаляються усі підлеглі таблиці

При введенні коректного запиту:



```
delete

Input table
"Hall"
Input condition or nothing to continue:
"HallID" = 1
All columns where "HallID" = 1 delete in table "Hall"
```

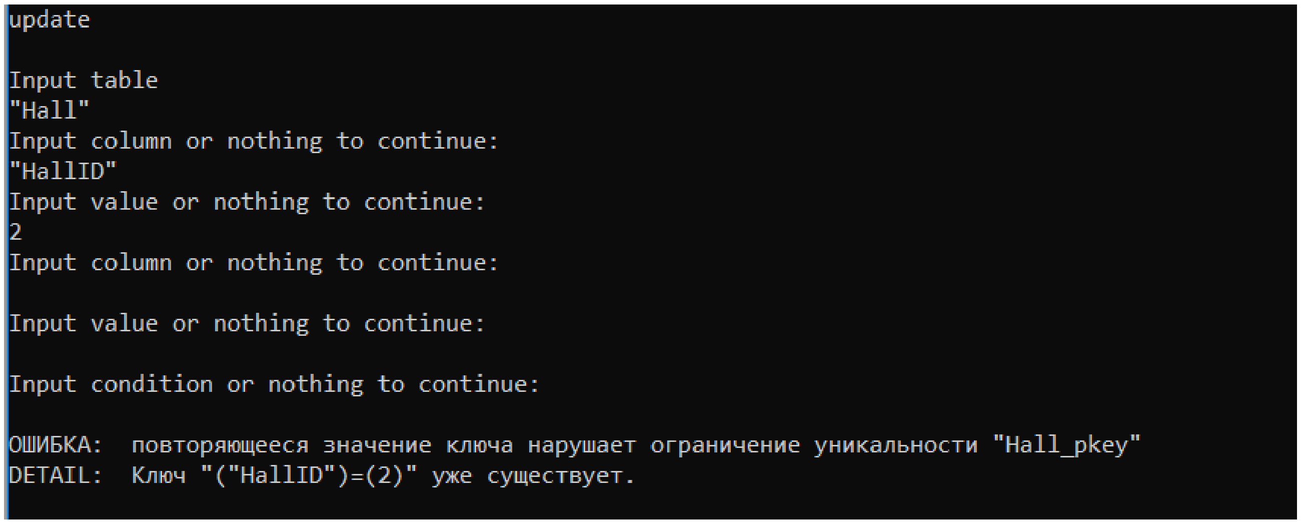
Валідація розписується в Help



```
help

Input string example: 'example'
Input table or column example: "TableID"
Separator - ,

Input something to continue...
```

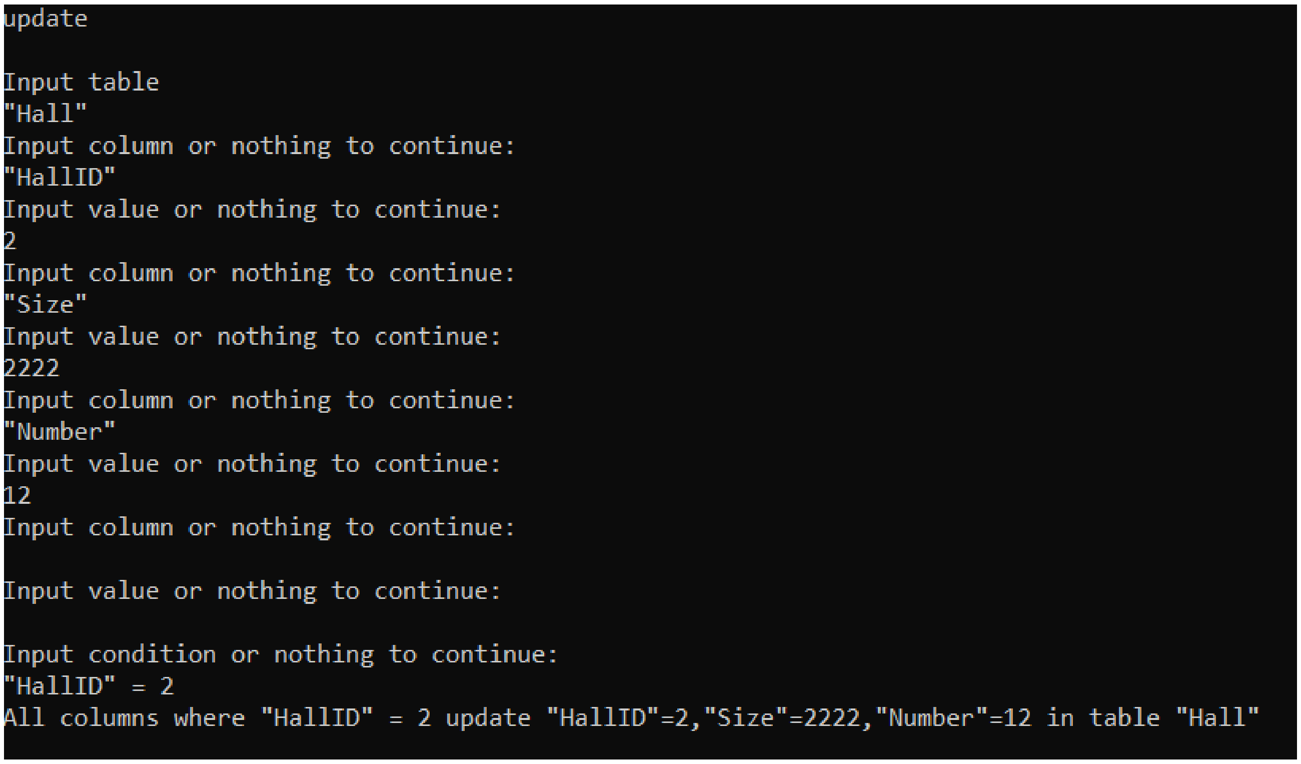
Update

При введенні зміни первинного ключа на ісеуючий:



```
update

Input table
"Hall"
Input column or nothing to continue:
"HallID"
Input value or nothing to continue:
2
Input column or nothing to continue:

Input value or nothing to continue:

Input condition or nothing to continue:

ОШИБКА:  повторяющееся значение ключа нарушает ограничение уникальности "Hall_pkey"
DETAIL:  Ключ "("HallID")=(2)" уже существует.
```

При введенні коректного запиту:

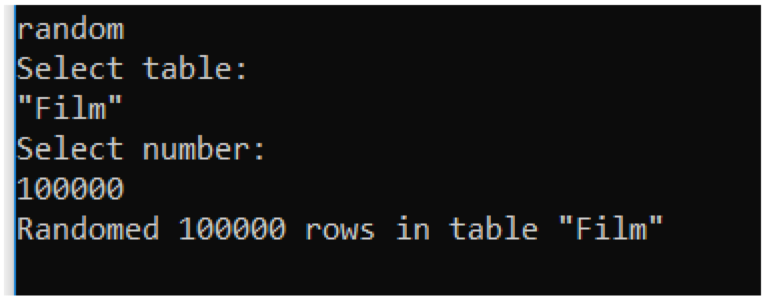


```
update

Input table
"Hall"
Input column or nothing to continue:
"HallID"
Input value or nothing to continue:
2
Input column or nothing to continue:
"Size"
Input value or nothing to continue:
2222
Input column or nothing to continue:
"Number"
Input value or nothing to continue:
12
Input column or nothing to continue:

Input value or nothing to continue:

Input condition or nothing to continue:
"HallID" = 2
All columns where "HallID" = 2 update "HallID"=2,"Size"=2222,"Number"=12 in table "Hall"
```

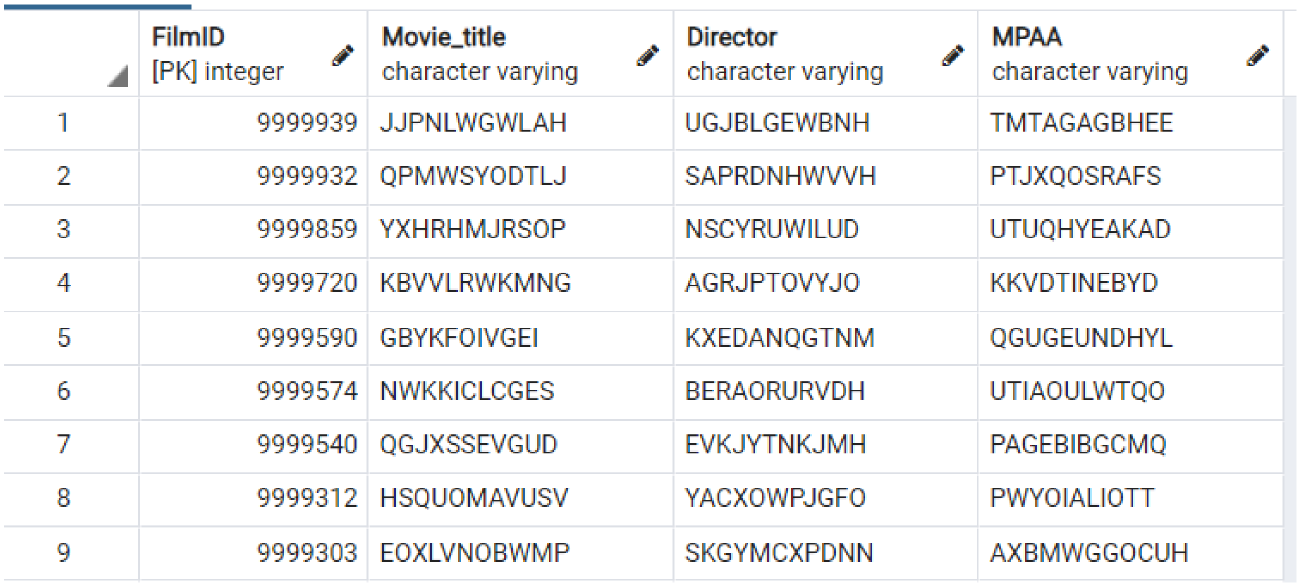
Завдання №2

Випадкове генерування даних доступно для усіх таблиць, але для таблиці “Performance/Hall” є обмеження тому, що зовнішні ключі є унікальними:

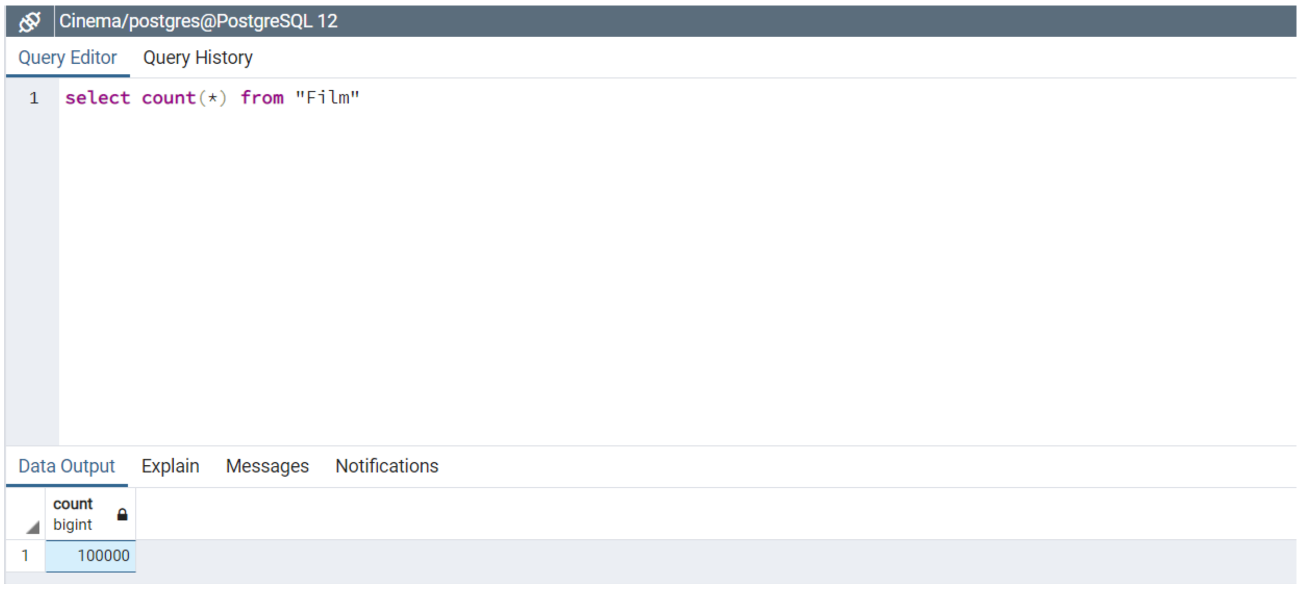
Випадкове генерування 100000 значень для таблиці “Film”



```
random
Select table:
"Film"
Select number:
100000
Randomed 100000 rows in table "Film"
```



	FilmID [PK] integer	Movie_title character varying	Director character varying	MPAA character varying
1	9999939	JJPNLWGWLAH	UGJBLGEWBNH	TMTAGAGBHEE
2	9999932	QPMWSYODTLJ	SAPRDNHWVVH	PTJXQOSRAFS
3	9999859	YXHRHMJRSOP	NSCYRUWILUD	UTUQHYEAKAD
4	9999720	KBVVLRWKMNG	AGRJPTOVYJO	KKVDTINEBYD
5	9999590	GBYKFOIVGEI	KXEDANQGTNM	QGUGEUNDHYL
6	9999574	NWKKICLCGES	BERAORURVDH	UTIAOULWTQO
7	9999540	QGJXSSEVGUD	EVKJYTNKJMH	PAGEBIBGCMQ
8	9999312	HSQUOMAVUSV	YACXOWPJGFO	PWYOIALIOTT
9	9999303	EOXLVNOBWMP	SKGYMCXPDNN	AXBMWGGOCUH

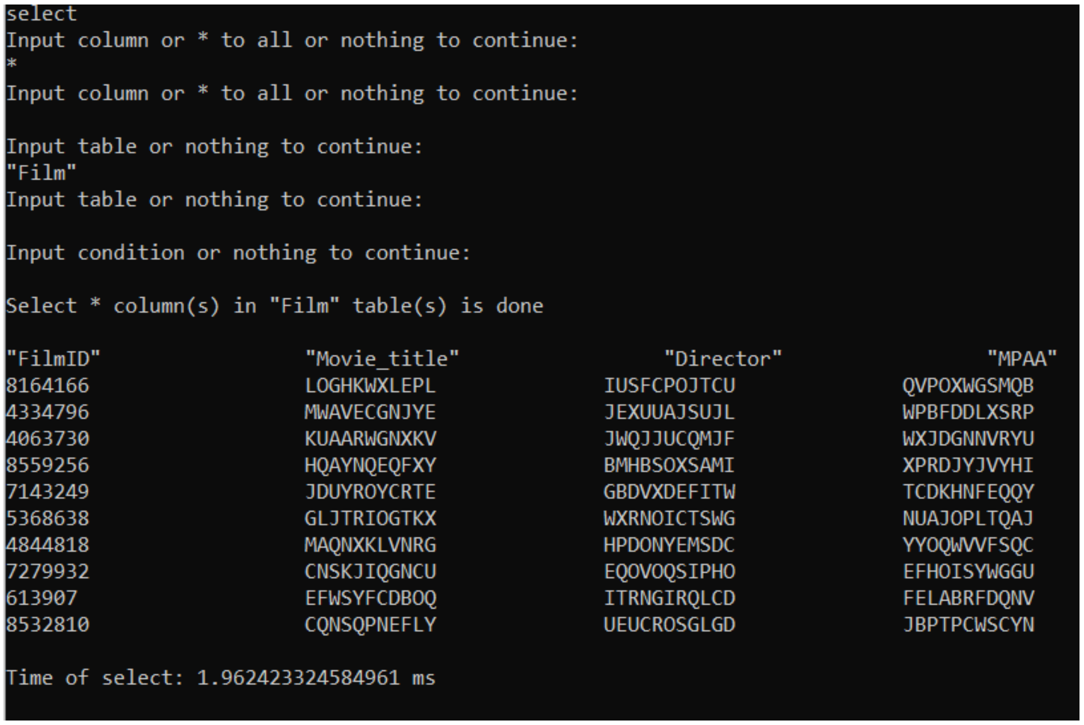


```
select count(*) from "Film"
```

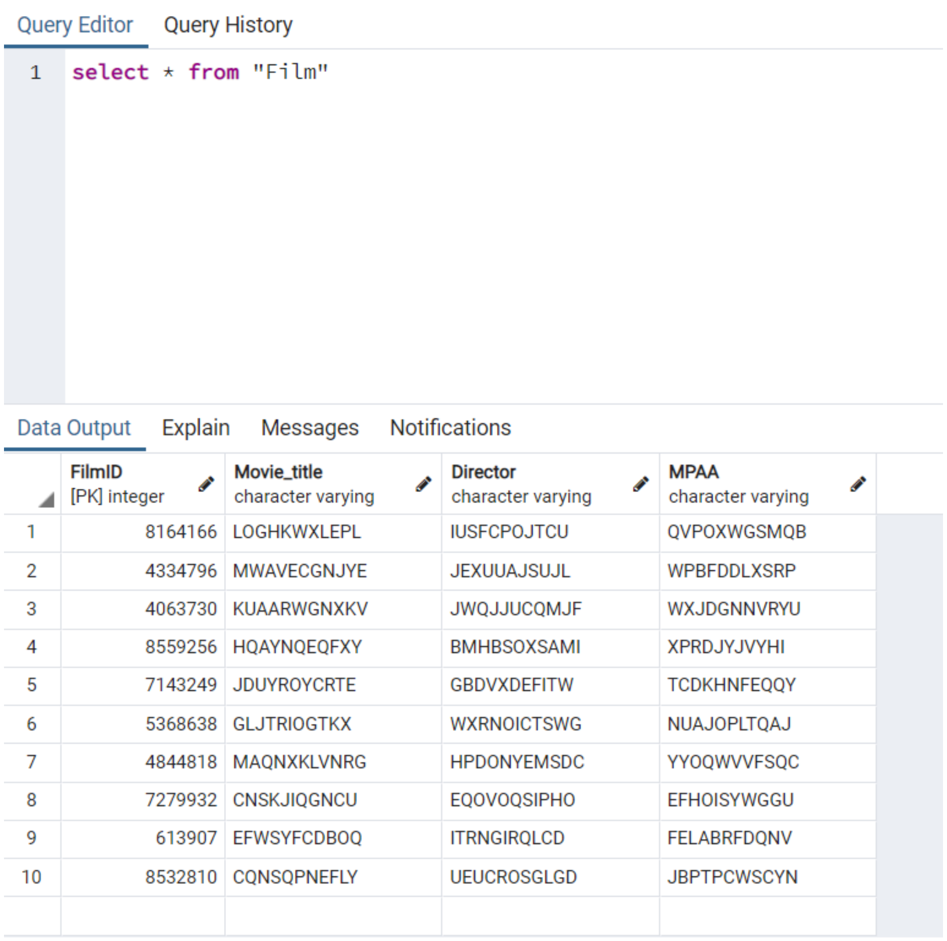
	count bigint
1	100000

Завдання №3

Select таблиці “Film”



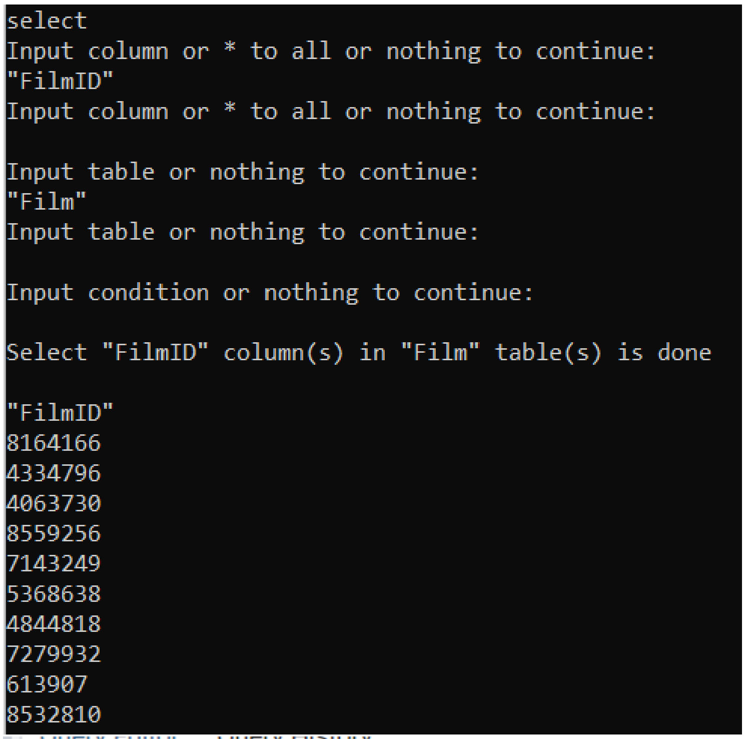
```
select
Input column or * to all or nothing to continue:
*
Input column or * to all or nothing to continue:

Input table or nothing to continue:
"Film"
Input table or nothing to continue:

Input condition or nothing to continue:

Select * column(s) in "Film" table(s) is done

"FilmID"                 "Movie_title"            "Director"                "MPAA"
8164166                  LOGHKWXLEPL              IUSFCPOJTCU              QVPOXWGSMQB
4334796                  MWAVECGNJYE              JEXUUAJSUJL              WPBFDDLXSRP
4063730                  KUAARWGNXKV              JWQJJUCQMJF              WXJDGNNVRYU
8559256                  HQAYNQEQFXY              BMHBSOXSAMI              XPRDJYJVYHI
7143249                  JDUYROYCRTE              GBDVXDEFITW              TCDKHNFEQQY
5368638                  GLJTRIOGTKX              WXRNOICTSWG              NUAJOPLTQAJ
4844818                  MAQNXKLVNRG              HPDONYEMSDC              YYOQWVVFSQC
7279932                  CNSKJIQGNCU              EQOVOQSIPHO              EFHOISYWGGU
613907                   EFWSYFCDBOQ              ITRNGIRQLCD              FELABRFDQNV
8532810                  CQNSQPNEFLY              UEUCROSGLGD              JBPTPCWSCYN

Time of select: 1.962423324584961 ms
```



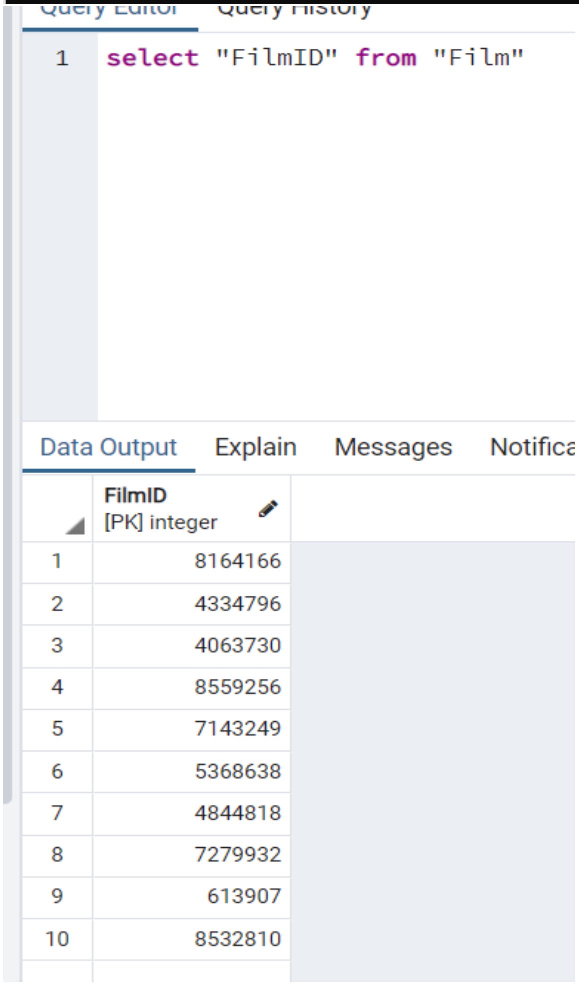
```
select * from "Film"
```

	FilmID [PK] integer	Movie_title character varying	Director character varying	MPAA character varying
1	8164166	LOGHKWXLEPL	IUSFCPOJTCU	QVPOXWGSMQB
2	4334796	MWAVECGNJYE	JEXUUAJSUJL	WPBFDDLXSRP
3	4063730	KUAARWGNXKV	JWQJJUCQMJF	WXJDGNNVRYU
4	8559256	HQAYNQEQFXY	BMHBSOXSAMI	XPRDJYJVYHI
5	7143249	JDUYROYCRTE	GBDVXDEFITW	TCDKHNFEQQY
6	5368638	GLJTRIOGTKX	WXRNOICTSWG	NUAJOPLTQAJ
7	4844818	MAQNXKLVNRG	HPDONYEMSDC	YYOQWVVFSQC
8	7279932	CNSKJIQGNCU	EQOVOQSIPHO	EFHOISYWGGU
9	613907	EFWSYFCDBOQ	ITRNGIRQLCD	FELABRFDQNV
10	8532810	CQNSQPNEFLY	UEUCROSGLGD	JBPTPCWSCYN

Select однієї колонки



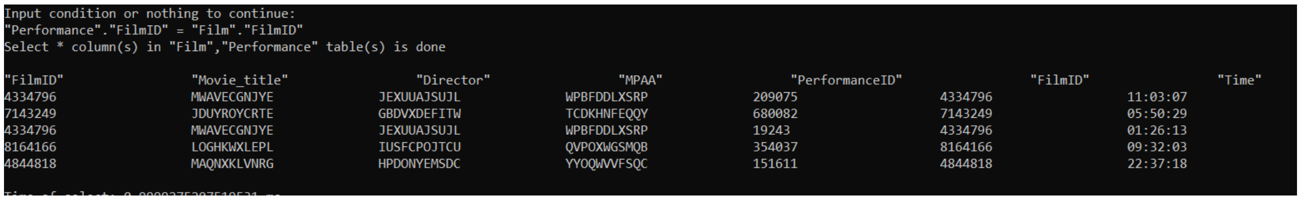
```
select
Input column or * to all or nothing to continue:
"FilmID"
Input column or * to all or nothing to continue:

Input table or nothing to continue:
"Film"
Input table or nothing to continue:

Input condition or nothing to continue:

Select "FilmID" column(s) in "Film" table(s) is done

"FilmID"
8164166
4334796
4063730
8559256
7143249
5368638
4844818
7279932
613907
8532810
```



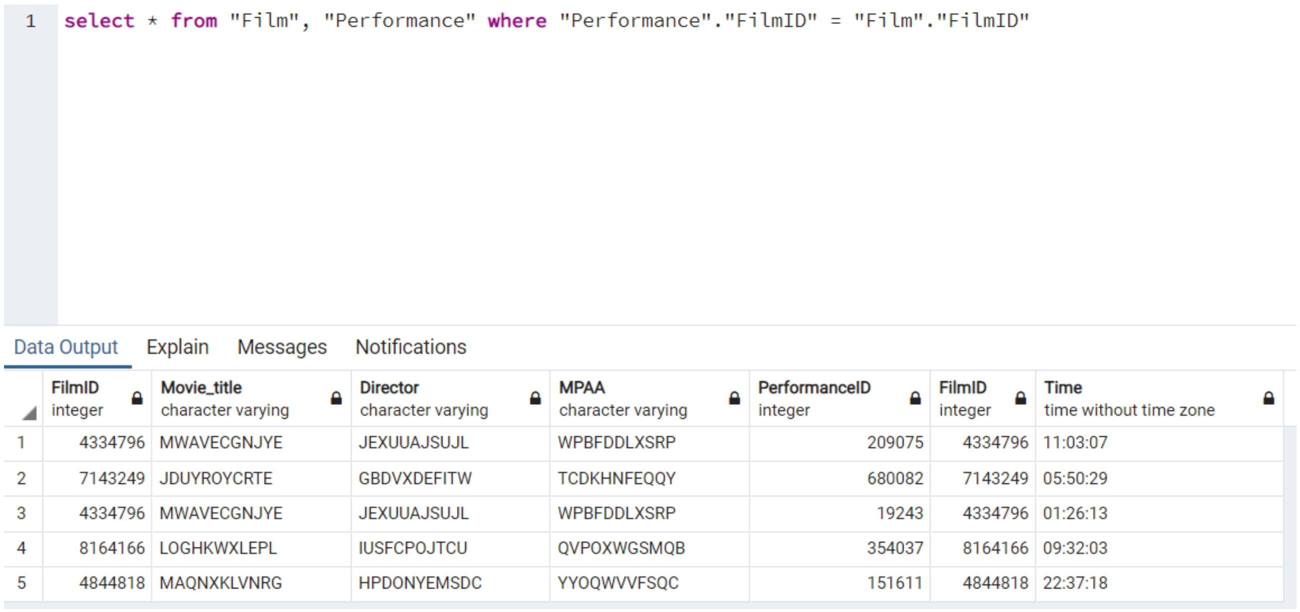
```
select "FilmID" from "Film"
```

	FilmID [PK] integer
1	8164166
2	4334796
3	4063730
4	8559256
5	7143249
6	5368638
7	4844818
8	7279932
9	613907
10	8532810

Select з двох таблиць за умовою



```
Input condition or nothing to continue:
"Performance"."FilmID" = "Film"."FilmID"
Select * column(s) in "Film","Performance" table(s) is done

"FilmID"                "Movie_title"              "Director"             "MPAA"                 "PerformanceID"              "FilmID"                 "Time"
4334796                 MWAVECGNJYE             JEXUUAJSUJL             WPBFDDLXSRP             209075                  4334796                 11:03:07
7143249                 JDUYROYCRTE             GBDVXDEFITW             TCDKHNFEQQY             680082                  7143249                 05:50:29
4334796                 MWAVECGNJYE             JEXUUAJSUJL             WPBFDDLXSRP             19243                   4334796                 01:26:13
8164166                 LOGHKWXLEPL             IUSFCPOJTCU             QVPOXWGSMQB             354037                  8164166                 09:32:03
4844818                 MAQNXKLVNRG             HPDONYEMSDC             YYOQWVVFSQC             151611                  4844818                 22:37:18
```



```
select * from "Film", "Performance" where "Performance"."FilmID" = "Film"."FilmID"
```

Data Output Explain Messages Notifications

	FilmID integer	Movie_title character varying	Director character varying	MPAA character varying	PerformanceID integer	FilmID integer	Time time without time zone
1	4334796	MWAVECGNJYE	JEXUUAJSUJL	WPBFDDLXSRP	209075	4334796	11:03:07
2	7143249	JDUYROYCRTE	GBDVXDEFITW	TCDKHNFEQQY	680082	7143249	05:50:29
3	4334796	MWAVECGNJYE	JEXUUAJSUJL	WPBFDDLXSRP	19243	4334796	01:26:13
4	8164166	LOGHKWXLEPL	IUSFCPOJTCU	QVPOXWGSMQB	354037	8164166	09:32:03
5	4844818	MAQNXKLVNRG	HPDONYEMSDC	YYOQWVVFSQC	151611	4844818	22:37:18

backend.py

```python
from psycopg2 import sql

def insert(cur, table, columns, values):
    cur.execute(sql.SQL("insert into " + table + " (" + columns + ") values " + values ))

def update(cur, table, set, condition):
    cur.execute(sql.SQL("update " + table + " set " + set + " where " + condition))

def delete(cur, table, condition):
    cur.execute(sql.SQL("delete from " + table + " where " + condition))

def str_rand_len(n):
    str = "chr(trunc(65 + random() * 25)::int)"
    i = 1
    for i in range(n):
        str += "|| chr(trunc(65 + random() * 25)::int)"
    return str

def random_one_film(cur, i, conn):
    try:
        cur.execute(sql.SQL("insert into \"Film\""  + " (\"FilmID\", \"Movie_title\", \"Director\", \"MPAA\") values (trunc(random()*10000000)::int," + str_rand_len(10) + ","
        return i + 1
    except:
        conn.rollback()
        return i

def random_film(cur, conn, n):
    i = 0
    while i < n:
        i = random_one_film(cur, i, conn)
        conn.commit()


def random_one_hall(cur, i, conn):
    try:
        cur.execute(sql.SQL("insert into \"Hall\"" + " (\"HallID\", \"Size\", \"Number\") values (trunc(random()*100000)::int, trunc(random()*1000000)::int, trunc(random()*100
         return i + 1
     except:
         conn.rollback()
         return i

 def random_hall(cur, conn, n):
     i = 0
     while i < n:
         i = random_one_hall(cur, i, conn)
         conn.commit()

 def random_one_performance(cur, i, conn):
     try:
         cur.execute(sql.SQL("insert into \"Performance\"" + " (\"PerformanceID\", \"FilmID\", \"Time\") values (trunc(random()*1000000)::int, (SELECT \"FilmID\" FROM \"Film\"
         return i + 1
     except Exception as e:
         conn.rollback()
         return i

 def random_performance(cur, conn, n):
     i = 0
     while i < n:
         i = random_one_performance(cur, i, conn)
         conn.commit()

 def random_one_performance_hall(cur, i, conn):
     try:
         cur.execute(sql.SQL("insert into \"Performance/Hall\"" + " (\"PerformanceHallID\", \"PerformanceID\", \"HallID\") values (trunc(random()*1000000)::int, (SELECT \"Perf
         return i + 1
     except Exception as e:
         conn.rollback()
         return i

 def random_performance_hall(cur, conn, n):
     i = 0
    while i < n:
        i = random_one_performance_hall(cur, i, conn)
        conn.commit()

def random_one_ticket(cur, i, conn):
    try:
        cur.execute(sql.SQL("insert into \"Ticket\"" + " (\"TicketID\", \"Seat\", \"Row\", \"PerformanceHallID\") values (trunc(random()*1000000)::int, trunc(random()*1000000)
        return i + 1
    except Exception as e:
        conn.rollback()
        return i

def random_ticket(cur, conn, n):
    i = 0
    while i < n:
        i = random_one_ticket(cur, i, conn)
        conn.commit()

def select(cur, columns, table, condition):
    cur.execute(sql.SQL("select " + columns + " from " + table + " where " + condition))
```

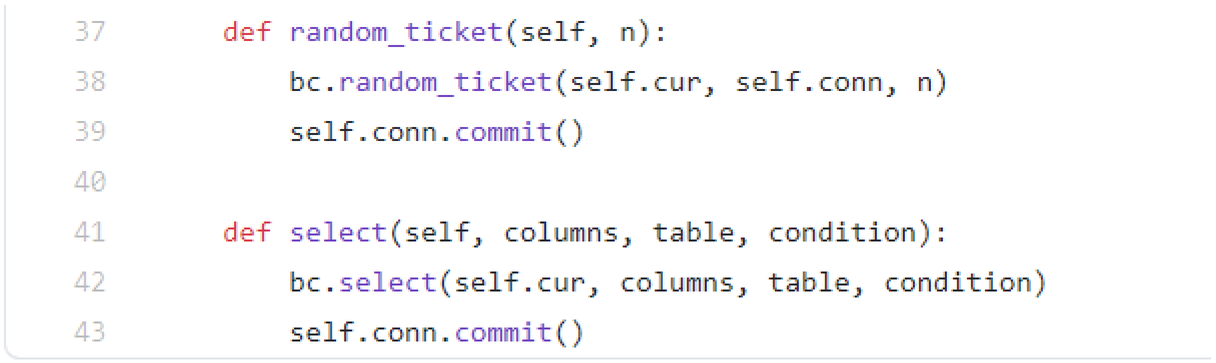
Controller.py

```
import backend as bc
import model as md
import view as vw
import psycopg2
import time

class Controller(object):

    def __init__(self, model, view):
        self.model = model
        self.view = view

    def insert(self, table, columns, values):
        try:
            self.model.insert(table, columns, values)
            self.view.display_insert(table, columns, values)
        except Exception as e:
            print("can't insert into {0} columns {1} values {2}".format(table, columns, values))
            print(e)
            self.model.conn.rollback()

    def update(self, table, set, condition):
        try:
            self.model.update(table, set, condition)
            self.view.display_update(table, set, condition)
        except Exception as e:
            print(e)
            self.model.conn.rollback()

    def delete(self, table, condition):
        try:
            if table == "\"Film\"":
                self.delete_film(table, condition)
            elif table == "\"Performance\"":
                self.delete_performance(table, condition)
            elif table == "\"Hall\"":
                self.delete_hall(table, condition)
            elif table == "\"Performance/Hall\"":
                self.delete_performance_hall(table, condition)
            elif table == "\"Ticket\"":
                self.delete_ticket(table, condition)
            #self.model.delete(table, condition)
            self.view.display_delete(table, condition)
        except Exception as e:
            if condition == "\'t\'":
                print("can't delete table {0}".format(table))
            else:
                print("can't delete table in {0} with condition {1}".format(table, condition))
            print(e)
            self.model.conn.rollback()

    def delete_film(self, table, condition):
        try:
            self.delete_performance("\"Performance\"", "\"FilmID\" in (select \"FilmID\" from \"Film\" where " + condition + ")")
            self.model.delete(table, condition)
        except Exception as e:
            print(e)
            self.model.conn.rollback()

    def delete_hall(self, table, condition):
        try:
            self.delete_performance_hall("\"Performance/Hall\"", "\"HallID\" in (select \"HallID\" from \"Hall\" where " + condition + ")")
            self.model.delete(table, condition)
        except Exception as e:
            print(e)
            self.model.conn.rollback()

    def delete_performance(self, table, condition):
        try:
            self.delete_performance_hall("\"Performance/Hall\"", "\"PerformanceID\" in (select \"PerformanceID\" from \"Performance\" where " + condition + ")")
            self.model.delete(table, condition)

        except Exception as e:
            print(e)
            self.model.conn.rollback()

    def delete_performance_hall(self, table, condition):
        try:
            self.delete_ticket("\"Ticket\"", "\"PerformanceHallID\" in (select \"PerformanceHallID\" from \"Performance/Hall\" where " + condition + ")")
            self.model.delete(table, condition)
        except Exception as e:
            print(e)
            self.model.conn.rollback()

    def delete_ticket(self, table, condition):
        try:
            self.model.delete(table, condition)
        except Exception as e:
            print(e)
            self.model.conn.rollback()

    def select(self, columns, table, condition, columns_all):
        try:
            Time = time.time()
            self.model.select(columns, table, condition)
            Time = time.time() - Time
            self.view.display_select(columns_all, table, self.model.cur, Time, columns)
        except Exception as e:
            print(e)
            self.model.conn.rollback()

    def random_table(self, table, n):
        try:
            if table == "\"Film\"":
                self.model.random_film(n)
            elif table == "\"Performance\"":
                self.model.random_performance(n)
            elif table == "\"Hall\"":
                self.model.random_hall(n)
            elif table == "\"Performance/Hall\"":
                self.model.random_performance_hall(n)
            elif table == "\"Ticket\"":
                self.model.random_ticket(n)
            #self.model.delete(table, condition)
            self.view.display_random(table, n)
        except Exception as e:
            print(e)
            self.model.conn.rollback()

def table_to_columns(table):
    n = 17
    if table == "\"Film\"":
        return "\"FilmID\"" + " " * n + "\"Movie_title\"" + " " * n + "\"Director\"" + " " * n + "\"MPAA\"" + " " * n
    elif table == "\"Hall\"":
        return "\"HallID\"" + " " * n + "\"Size\"" + " " * n + "\"Number\"" + " " * n
    elif table == "\"Performance\"":
        return "\"PerformanceID\"" + " " * n + "\"FilmID\"" + " " * n + "\"Time\"" + " " * n
    elif table == "\"Performance/Hall\"":
        return "\"PerformanceHallID\"" + " " * n + "\"PerformanceID\"" + " " * n + "\"HallID\"" + " " * n
    elif table == "\"Ticket\"":
        return "\"TicketID\"" + " " * n + "\"PerformanceHallID\"" + " " * n + "\"Seat\"" + " " * n + "\"Row\"" + " " * n
    else:
        return ""

def menu():
    conn = psycopg2.connect(dbname="Cinema", user="postgres", password="e28n3t0")
    cur = conn.cursor()
```

```python
    c = Controller(md.Model(cur, conn), vw.View())
    work = True
    previos_menu_type = "MAIN"
    menu_type = "MAIN"
    while(work):
        if menu_type == "MAIN":
            print("\nTables")
            print("Insert")
            print("Update")
            print("Delete")
            print("Random")
            print("Select")
            print("Help")
            print("Exit\n")
            previos_menu_type = "MAIN"
            menu_type = input().upper()
        elif menu_type == "TABLES":
            print("\nFilm:")
            print("FilmID - int; Movie_title - string; Director - string; MPAA - string")
            print("Performance:")
            print("PerformanceID - int; FilmID - int; Time - time(23:59:59)")
            print("Hall:")
            print("HallID - int; Size - int; Number - int")
            print("Performance/Hall:")
            print("PerformanceHallID - int; PerformanceID - int; HallID - int")
            print("Ticket:")
            print("TicketID - int; Seat - int; Row - int; PerformanceHallID - int")
            print("\nInput something to continue...\n")
            input()
            menu_type = "MAIN"
        elif menu_type == "INSERT":
            columns = ""
            values = ""
            value = " "
            print("\nInput table continue:\n")
            table = input()
            print("\nInput columns(separator - ,)")
            columns = input()
            while len(value) != 0:
                print("Input values or nothing to continue(separator - ,):")
                value = input()
                if len(value) != 0:
                    value = "(" + value + ")"
                    if len(values) != 0:
                        values += ","
                values += value
            c.insert(table, columns, values)
            print("\nInput something to continue...\n")
            input()
            menu_type = "MAIN"
        elif menu_type == "UPDATE":
            set = ""
            str = " "
            cond = ""
            print("\nInput table")
            table = input()
            while len(str) != 0:
                print("Input column or nothing to continue:")
                str = input()
                if len(str) != 0:
                    str += "="
                    if len(set) != 0:
                        set += ","
                print("Input value or nothing to continue:")
                str += input()
                set += str
            print("Input condition or nothing to continue:")
            cond = input()
            if len(cond) == 0:
                cond = "\'t\'"
            c.update(table, set, cond)
            menu_type = "MAIN"
        elif menu_type == "DELETE":
            print("\nInput table")
            table = input()
            print("Input condition or nothing to continue:")
            cond = input()
            if len(cond) == 0:
                cond = "\'t\'"
            c.delete(table, cond)
            menu_type = "MAIN"
        elif menu_type == "RANDOM":
            print("Select table:")
            table = input()
            try:
                print("Select number:")
                n = int(input())
                c.random_table(table, n)
            except:
                print("Its not number")

            menu_type = "MAIN"
        elif menu_type == "SELECT":
            tables = ""
            table = " "
            columns = ""
            column = " "
            columns_all = ""
            while len(column) != 0:
                print("Input column or * to all or nothing to continue:")
                column = input()
                if len(column) != 0 and len(tables) != 0:
                    columns += ","
                columns += column
            while len(table) != 0:
                print("Input table or nothing to continue:")
                table = input()
                if len(table) != 0:
                    if len(tables) != 0:
                        tables += ","
                    if columns == "*":
                        columns_all += table_to_columns(table)
                    else:
                        columns_all = columns

                tables += table

            print("Input condition or nothing to continue:")
            cond = input()
            if len(cond) == 0:
                cond = "\'t\'"

            c.select(columns, tables, cond, columns_all)
            menu_type = "MAIN"
        elif menu_type == "HELP":
            print("\nInput string example: \'example\'")
            print("Input table or column example: \"TableID\"")
            print("Separator - ,")
            print("\nInput something to continue...\n")
            input()
            menu_type = "MAIN"
        elif menu_type == "EXIT":
            work = False
        else:

            menu_type = previos_menu_type
    cur.close()
    conn.close()

menu()
```

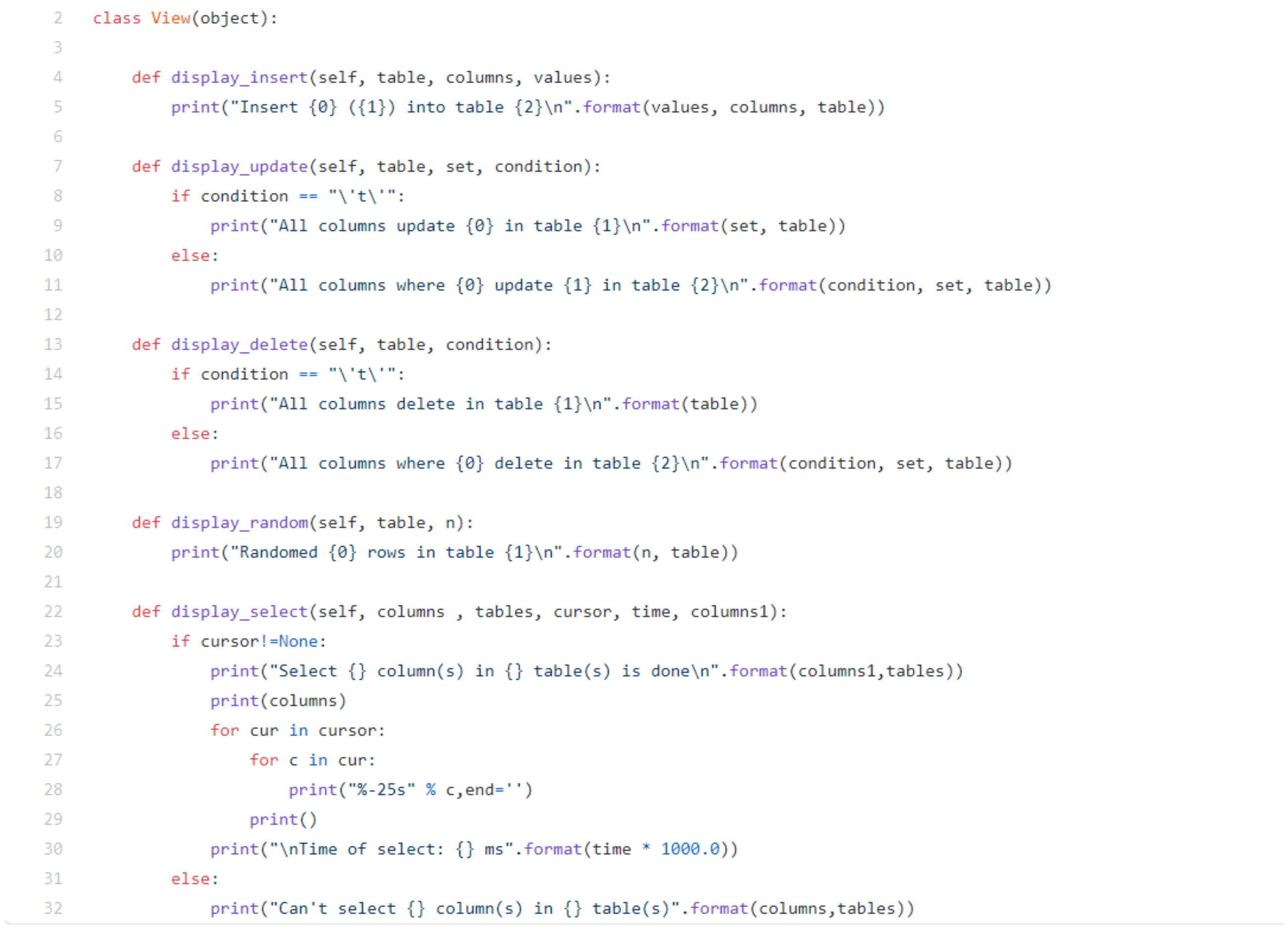
model.py

```python
import backend as bc

class Model(object):

    def __init__(self, input_cur, input_conn):
        self.cur = input_cur
        self.conn = input_conn

    def insert(self, table, columns, values):
        bc.insert(self.cur, table, columns, values)
        self.conn.commit()

    def update(self, table, set, condition):
        bc.update(self.cur, table, set, condition)
        self.conn.commit()

    def delete(self, table, condition):
        bc.delete(self.cur, table, condition)
        self.conn.commit()

    def random_film(self, n):
        bc.random_film(self.cur, self.conn, n)
        self.conn.commit()

    def random_hall(self, n):
        bc.random_hall(self.cur, self.conn, n)
        self.conn.commit()

    def random_performance(self, n):
        bc.random_performance(self.cur, self.conn, n)
        self.conn.commit()

    def random_performance_hall(self, n):
        bc.random_performance_hall(self.cur, self.conn, n)
        self.conn.commit()

```



```python
    def random_ticket(self, n):
        bc.random_ticket(self.cur, self.conn, n)
        self.conn.commit()

    def select(self, columns, table, condition):
        bc.select(self.cur, columns, table, condition)
        self.conn.commit()
```

view.py



```python
class View(object):

    def display_insert(self, table, columns, values):
        print("Insert {0} ({1}) into table {2}\n".format(values, columns, table))

    def display_update(self, table, set, condition):
        if condition == "\'t\'":
            print("All columns update {0} in table {1}\n".format(set, table))
        else:
            print("All columns where {0} update {1} in table {2}\n".format(condition, set, table))

    def display_delete(self, table, condition):
        if condition == "\'t\'":
            print("All columns delete in table {1}\n".format(table))
        else:
            print("All columns where {0} delete in table {2}\n".format(condition, set, table))

    def display_random(self, table, n):
        print("Randomed {0} rows in table {1}\n".format(n, table))

    def display_select(self, columns , tables, cursor, time, columns1):
        if cursor!=None:
            print("Select {} column(s) in {} table(s) is done\n".format(columns1,tables))
            print(columns)
            for cur in cursor:
                for c in cur:
                    print("%-25s" % c,end='')
                print()
            print("\nTime of select: {} ms".format(time * 1000.0))
        else:
            print("Can't select {} column(s) in {} table(s)".format(columns,tables))
```