



НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ  
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені Ігоря Сікорського»  
ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ  
**Кафедра системного програмування та спеціалізованих комп'ютерних систем**

**Розрахунково-графічна робота**

з дисципліни

**«Бази даних та засоби управління»**

Тема роботи: «Створення додатку бази даних, орієнтованого на взаємодію з СУБД PostgreSQL»

Виконав: студент групи КВ-11  
Карлаш Іван

Telegram: @jnguar

Київ – 2023

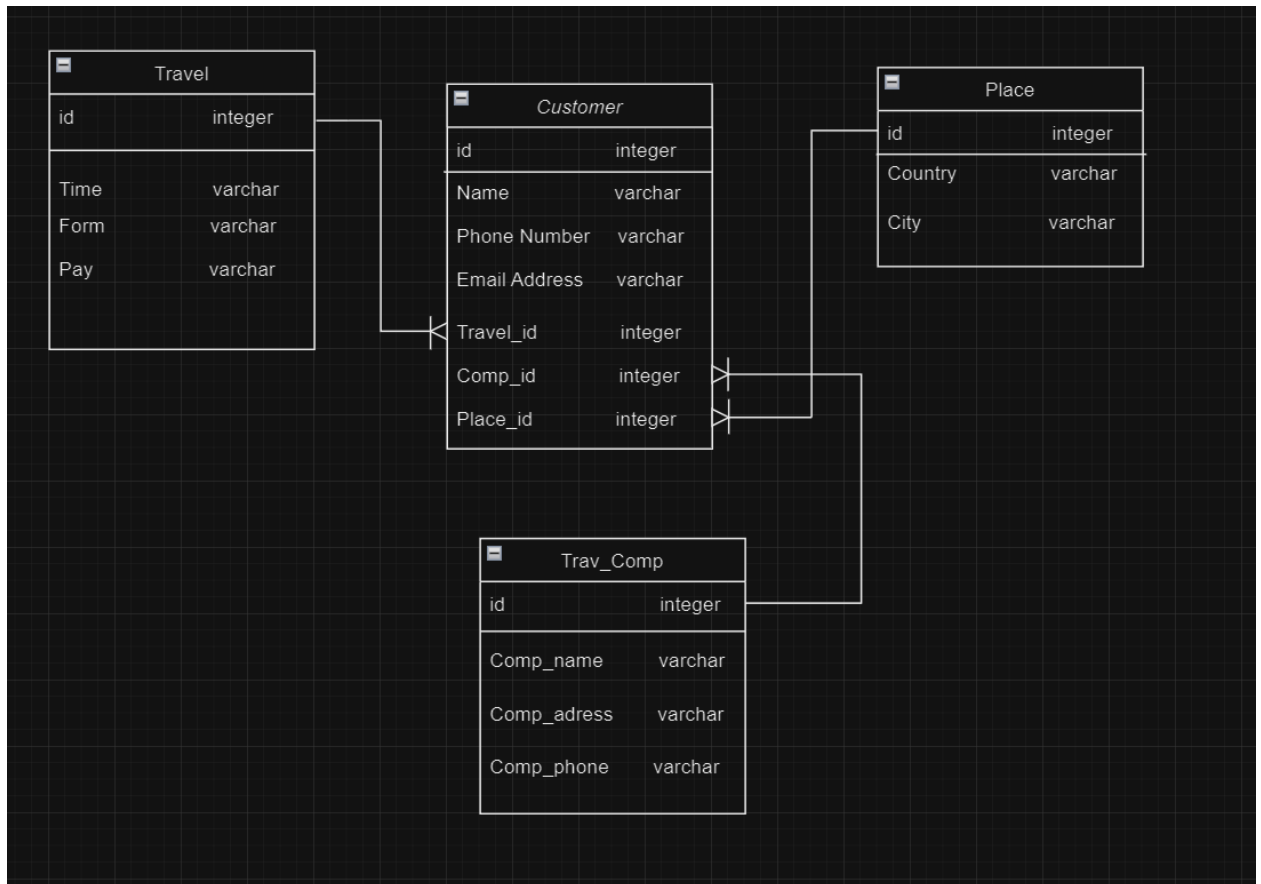
**Метою роботи** є здобуття вмінь програмування прикладних додатків баз даних PostgreSQL.

Загальне завдання роботи полягає у наступному:

1. Реалізувати функції перегляду, 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-поданняконтролер).

Посилання на Github: [https://github.com/vaniapelman/DB\\_lab](https://github.com/vaniapelman/DB_lab)

Скріншот розробленої моделі «сутність-зв'язок»:



Опис сутностей:

1) Сутність «Customer» (покупець) має 7 атрибутів:

- Id – ідентифікатор покупця, змінюється відповідно кількості покупців;
- Name – текстове поле, що позначає ім'я покупця;
- Phone Number – текстове поле, що позначає номер телефону покупця;
- Email Address – текстове поле, що позначає емеїл адресу;
- Travel\_id – зв'язок з сутністю Travel;
- Comp\_id – зв'язок з сутністю Trav\_Comp;
- Place\_id - зв'язок з сутністю Place;

2) Сутність «Travel» (подорож) має 4 атрибути:

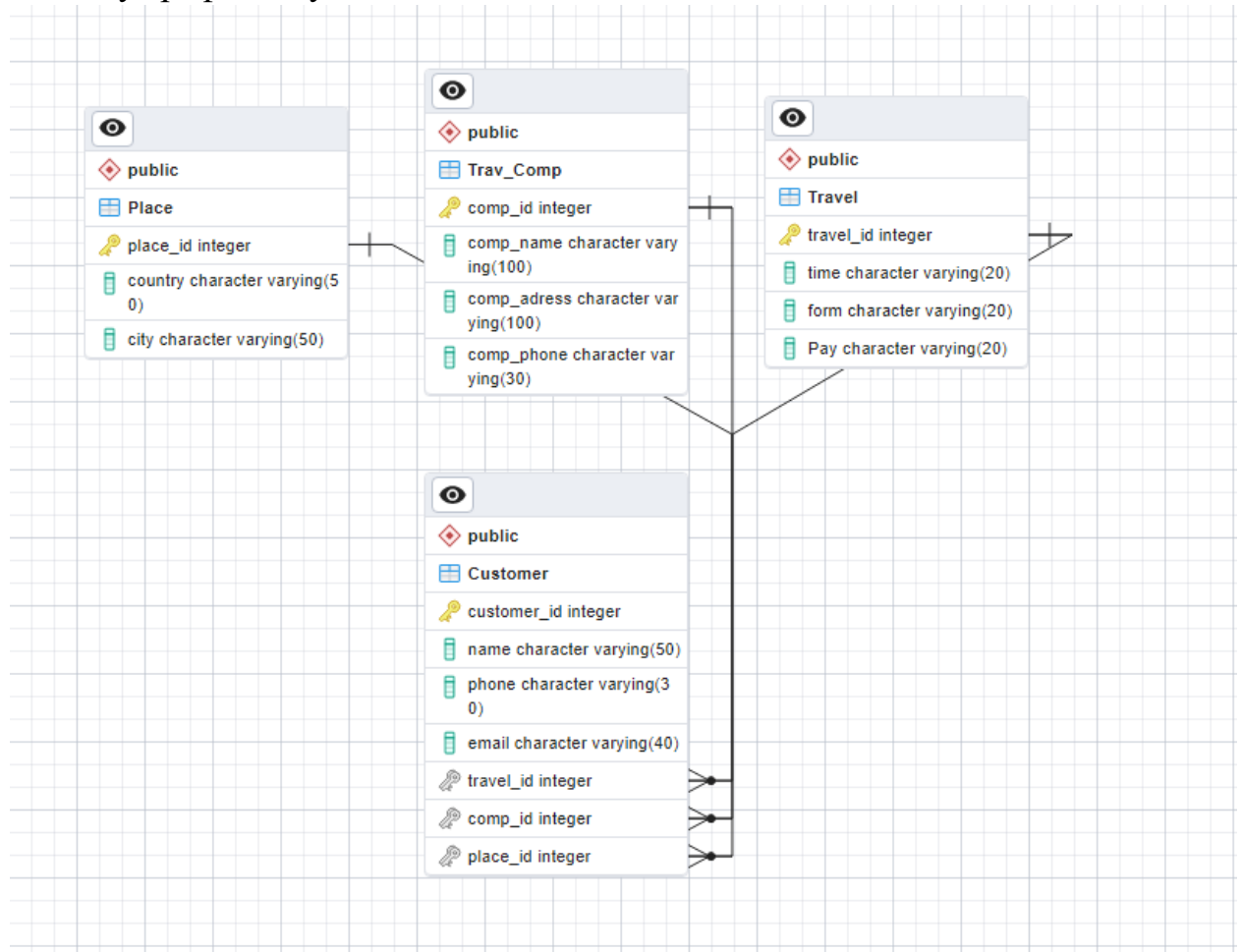
- Id – унікальний ідентифікатор поїздки;
- Time – час відправлення;
- Form – Якість поїздки(преміум, звичайна);
- Pay – Статус оплати, оплачено, чи ні;

3) Сутність «Trav\_comp» (компанія транспортування) має 4 атрибути:

- Id – ідентифікатор транспортної компанії;
- Comp\_name – текстове поле, з назвою компанії;
- Comp\_adress – текстове поле з адресою компанії;

- Comp\_phone – текстове поле з номером телефону компанії;
- 4) Сутність «Place» (місце подорожі) має 3 атрибути:
- Id – ідентифікатор місця подорожі;
  - Country – текстове поле, назва країни;
  - City – текстове поле, назва міста;

Схема у графічному вигляді:



### Меню користувача:

```
Menu:  
1.Add line  
2.Add random line(for "customers")  
3.Show table  
4.Update line  
5.Delete line  
6.Search  
7.Exit  
Choose:
```

```
Tables:  
1.Customer  
2.Trav_comp  
3.Travel  
4.Place  
5.Back to menu  
Choose table:
```

### В меню 7 пунктів:

1. Add line (додає рядок, в таблицю яку оберете)
2. Add random (додає рядок рандомних значень в customer)
3. Show table(показує таблицю яку оберете)
4. Update line(оновлює данні обраного рядка таблиці)
5. Delete line(видаляє рядок)
6. Search(виконує функції пошуку/впорядкування)
7. Exit(вихід з програми)

### Для розробки використовувалось:

Середовище для відлагодження SQL-запитів до бази даних – PgAdmin4.

Мова програмування – Python 3.10.

Середовище розробки програмного забезпечення – PyCharm.

Бібліотека взаємодії з PostgreSQL - psycopg2.

## Частина №1

### ■ Внесення даних в customer:

	customer_id [PK] integer	name character varying (50)	phone character varying (30)	email character varying (40)	travel_id integer	comp_id integer	place_id integer
1	1	Bob Slam	976453276	bob@gmail.com	2	1	3
2	2	Maria Lonsey	637562834	lonsi_sas@gmail.com	3	2	5
3	3	Ilon Conra	964539821	g123kh@gmai.com	4	1	2
4	4	Rian Valdoro	638354286	samsobi@gmail.com	4	2	2
5	5	Lina Plaso	976245923	linopla@gmail.com	1	3	6

Menu:

```
1.Add line
2.Add random line(for "customers")
3.Show table
4.Update line
5.Delete line
6.Search
7.Exit
Choose: 1
```

Tables:

```
1.Customer
2.Trav_comp
3.Travel
4.Place
5.Back to menu
Choose table: 1
```

Adding customer:

```
Enter customer name: Lola Alol
Enter customer phone: 975463425
Enter customer email: lolahr@gmail.com
Enter customer travel_id: 1
Enter customer comp_id: 2
Enter customer place_id: 4
Customer added successfully!
```

Таблиця customer після внесених змін:

	customer_id [PK] integer	name character varying (50)	phone character varying (30)	email character varying (40)	travel_id integer	comp_id integer	place_id integer
1	1	Bob Slam	976453276	bob@gmail.com	2	1	3
2	2	Maria Lonsey	637562834	lonsi_sas@gmail.com	3	2	5
3	3	Ilon Conra	964539821	g123kh@gmai.com	4	1	2
4	4	Rian Valdoro	638354286	samsobi@gmail.com	4	2	2
5	5	Lina Plaso	976245923	linopla@gmail.com	1	3	6
6	6	Lola Alol	975463425	lolahr@gmail.com	1	2	4

▪ Внесення даних в Trav\_comp:

	comp_id [PK] integer	comp_name character varying (100)	comp_address character varying (100)	comp_phone character varying (30)
1	1	Lufthansa	Keln, Germany	+380934568723
2	2	Eurostar	Paris, France	+380674563421
3	3	Eurolines	Brussels, Belgium	+380936740326

```
Tables:
1.Customer
2.Trav_comp
3.Travel
4.Place
5.Back to menu
Choose table: 2

Adding trcomp:
Enter trcomp comp_name: Lambada
Enter trcomp comp_address: Kyiv, Ukraine
Enter trcomp comp_phone: +380976543723
Trcomp added successfully!
```

Таблиця trcomp після внесення змін:

	comp_id [PK] integer	comp_name character varying (100)	comp_address character varying (100)	comp_phone character varying (30)
1	1	Lufthansa	Keln, Germany	+380934568723
2	2	Eurostar	Paris, France	+380674563421
3	3	Eurolines	Brussels, Belgium	+380936740326
4	4	Lambada	Kyiv, Ukraine	+380976543723

▪ **Внесення даних в travel:**

	travel_id [PK] integer	time character varying (20)	form character varying (20)	Pay character varying (20)
1	1	20:00	Premium	Yes
2	2	20:00	Premium	No
3	3	10:00	Casual	Yes
4	4	10:00	Casual	No

```

Tables:
1.Customer
2.Trav_comp
3.Travel
4.Place
5.Back to menu
Choose table: 3

Adding travel:
Enter travel time: 22:00
Enter travel form: Premium
Enter travel pay: Yes
Travel added successfully!




```

Таблиця travel після внесення змін:

	travel_id [PK] integer	time character varying (20)	form character varying (20)	pay character varying (20)
1	1	20:00	Premium	Yes
2	2	20:00	Premium	No
3	3	10:00	Casual	Yes
4	4	10:00	Casual	No
5	5	22:00	Premium	Yes



■ **Внесення даних в Place:**

	place_id [PK] integer 	country character varying (50) 	city character varying (50) 
1	1	Poland	Warsaw
2	2	Poland	Krakow
3	3	Italy	Rome
4	4	Italy	Milan
5	5	France	Paris
6	6	Germany	Berlin




```

Tables:
1.Customer
2.Trav_comp
3.Travel
4.Place
5.Back to menu
Choose table: 4

Adding place:
Enter place country: Latvia
Enter place city: Riga
Place added successfully!

```

Таблиця place після внесення змін:

	place_id [PK] integer 	country character varying (50) 	city character varying (50) 
1	1	Poland	Warsaw
2	2	Poland	Krakow
3	3	Italy	Rome
4	4	Italy	Milan
5	5	France	Paris
6	6	Germany	Berlin
7	7	Latvia	Riga

## Перегляд даних:

### ▪ Таблиця customer:

```
Menu:
1.Add line
2.Add random line(for "customers")
3.Show table
4.Update line
5.Delete line
6.Search
7.Exit
Choose: 3

Tables:
1.Customer
2.Trav_comp
3.Travel
4.Place
5.Back to menu
Choose table: 1

customer:
ID: 6, Name: Lola AloI, Phone: 975463425, Email: lolahr@gmail.com, travel_id: 1, comp_id: 2, place_id: 4
ID: 1, Name: Bob Slam, Phone: 976453276, Email: bob@gmail.com, travel_id: 2, comp_id: 1, place_id: 3
ID: 2, Name: Maria Lonsey, Phone: 637562834, Email: lonsi_sas@gmail.com, travel_id: 3, comp_id: 2, place_id: 5
ID: 3, Name: Ilon Conra, Phone: 964539821, Email: g123Kh@gmai.com, travel_id: 4, comp_id: 1, place_id: 2
ID: 4, Name: Rian Valdoro, Phone: 638354286, Email: samsobi@gmail.com, travel_id: 4, comp_id: 2, place_id: 2
ID: 5, Name: Lina Plaso, Phone: 976245923, Email: linopla@gmail.com, travel_id: 1, comp_id: 3, place_id: 6
```

### ▪ Таблиця Trav\_comp:

```
Tables:
1.Customer
2.Trav_comp
3.Travel
4.Place
5.Back to menu
Choose table: 2

trcomps:
ID: 1, Name: Lufthansa, Address: Keln, Germany, Phone: +380934568723
ID: 2, Name: Eurostar, Address: Paris, France, Phone: +380674563421
ID: 3, Name: Eurolines, Address: Brussels, Belgium, Phone: +380936740326
ID: 4, Name: Lambada, Address: Kyiv, Ukraine, Phone: +380976543723
```

### ▪ Таблиця Travel:

```
Tables:
1.Customer
2.Trav_comp
3.Travel
4.Place
5.Back to menu
Choose table: 3

Travels:
ID: 1, Time: 20:00, Form: Premium, Pay: Yes
ID: 2, Time: 20:00, Form: Premium, Pay: No
ID: 3, Time: 10:00, Form: Casual, Pay: Yes
ID: 4, Time: 10:00, Form: Casual, Pay: No
ID: 5, Time: 22:00, Form: Premium, Pay: Yes
```

▪ **Таблица Place:**

```
Tables:
1.Customer
2.Trav_comp
3.Travel
4.Place
5.Back to menu
Choose table: 4

places:
ID: 1, Country: Poland, City: Warsaw
ID: 2, Country: Poland, City: Krakow
ID: 3, Country: Italy, City: Rome
ID: 4, Country: Italy, City: Milan
ID: 5, Country: France, City: Paris
ID: 6, Country: Germany, City: Berlin
ID: 7, Country: Latvia, City: Riga
```

## Редагування даних:

### ■ Таблиця Customer

	customer_id [PK] integer	name character varying (50)	phone character varying (30)	email character varying (40)	travel_id integer	comp_id integer	place_id integer
1	1	Bob Slam	976453276	bob@gmail.com	2	1	3
2	2	Maria Lonsey	637562834	lonsi_sas@gmail.com	3	2	5
3	3	Ilon Conra	964539821	g123kh@gmai.com	4	1	2
4	4	Rian Valdoro	638354286	samsobi@gmail.com	4	2	2
5	5	Lina Plaso	976245923	linopla@gmail.com	1	3	6
6	6	Lola Alol	975463425	lolahr@gmail.com	1	2	4

```
Menu:
1.Add line
2.Add random line(for "customers")
3.Show table
4.Update line
5.Delete line
6.Search
7.Exit
Choose: 4

Tables:
1.Customer
2.Trav_comp
3.Travel
4.Place
5.Back to menu
Choose table: 1

Updating customer:
Enter customer ID: 6
Enter customer name: Tot Otot
Enter customer phone: 675463865
Enter customer email: totttot@gmail.com
Enter customer travel_id: 1
Enter customer comp_id: 2
Enter customer place_id: 3
Customer updated successfully!
```

### Таблиця customer після редагування:

	customer_id [PK] integer	name character varying (50)	phone character varying (30)	email character varying (40)	travel_id integer	comp_id integer	place_id integer
1	1	Bob Slam	976453276	bob@gmail.com	2	1	3
2	2	Maria Lonsey	637562834	lonsi_sas@gmail.com	3	2	5
3	3	Ilon Conra	964539821	g123kh@gmai.com	4	1	2
4	4	Rian Valdoro	638354286	samsobi@gmail.com	4	2	2
5	5	Lina Plaso	976245923	linopla@gmail.com	1	3	6
6	6	Tot Otot	675463865	totttot@gmail.com	1	2	3

### ▪ Таблиця Trav\_comp

	comp_id [PK] integer	comp_name character varying (100)	comp_address character varying (100)	comp_phone character varying (30)
1	1	Lufthansa	Keln, Germany	+380934568723
2	2	Eurostar	Paris, France	+380674563421
3	3	Eurolines	Brussels, Belgium	+380936740326
4	4	Lambada	Kyiv, Ukraine	+380976543723

```

Tables:
1.Customer
2.Trav_comp
3.Travel
4.Place
5.Back to menu
Choose table: 2





Updating trcomp:
Enter trcomp ID: 1
Enter trcomp comp_name: Altion
Enter trcomp comp_address: Lviv, Ukraine
Enter trcomp comp_phone: +380976548723
Trcomp updated successfully!

```

Таблиця trav\_comp після редагування:

	comp_id [PK] integer	comp_name character varying (100)	comp_address character varying (100)	comp_phone character varying (30)
1	1	Altion	Lviv, Ukraine	+380976548723
2	2	Eurostar	Paris, France	+380674563421
3	3	Eurolines	Brussels, Belgium	+380936740326
4	4	Lambada	Kyiv, Ukraine	+380976543723

## ▪ Таблиця Travel

	travel_id [PK] integer 	time character varying (20) 	form character varying (20) 	pay character varying (20) 
1	1	20:00	Premium	Yes
2	2	20:00	Premium	No
3	3	10:00	Casual	Yes
4	4	10:00	Casual	No
5	5	22:00	Premium	Yes





```

Tables:
1.Customer
2.Trav_comp
3.Travel
4.Place
5.Back to menu
Choose table: 3

Updating travel:
Enter travel ID: 1
Enter travel time: 24:00
Enter travel form: Premium
Enter travel pay: Yes
Travel updated successfully!

```

Таблиця travel після редагування:

	travel_id [PK] integer 	time character varying (20) 	form character varying (20) 	pay character varying (20) 
1	1	24:00	Premium	Yes
2	2	20:00	Premium	No
3	3	10:00	Casual	Yes
4	4	10:00	Casual	No
5	5	22:00	Premium	Yes

## ■ Таблиця Place

	place_id [PK] integer	country character varying (50)	city character varying (50)
1	1	Poland	Warsaw
2	2	Poland	Krakow
3	3	Italy	Rome
4	4	Italy	Milan
5	5	France	Paris
6	6	Germany	Berlin
7	7	Latvia	Riga

Tables:

1.Customer

2.Trav\_comp

3.Travel

4.Place

5.Back to menu

Choose table: 4

Updating place:

Enter place ID: 1

Enter place country: *Ukrain*

Enter place city: *Lviv*

Place updated successfully!

Таблиця place після редагування:

	place_id [PK] integer	country character varying (50)	city character varying (50)
1	1	Ukrain	Lviv
2	2	Poland	Krakow
3	3	Italy	Rome
4	4	Italy	Milan
5	5	France	Paris
6	6	Germany	Berlin
7	7	Latvia	Riga

## Частина №2

Додамо рядок рандомних значень в customer:

	customer_id [PK] integer	name character varying (50)	phone character varying (30)	email character varying (40)	travel_id integer	comp_id integer	place_id integer
1	1	Bob Slam	976453276	bob@gmail.com	2	1	3
2	2	Maria Lonsey	637562834	lonsi_sas@gmail.com	3	2	5
3	3	Ilon Conra	964539821	g123kh@gmail.com	4	1	2
4	4	Rian Valdoro	638354286	samsobi@gmail.com	4	2	2
5	5	Lina Plaso	976245923	linopla@gmail.com	1	3	6
6	6	Tot Otot	675463865	totttot@gmail.com	1	2	3

Menu:

```
1.Add line
2.Add random line(for "customers")
3.Show table
4.Update line
5.Delete line
6.Search
7.Exit
Choose: 2
```

Adding random customer:

```
Enter the number: 1
Random fields added successfully!
```

	customer_id [PK] integer	name character varying (50)	phone character varying (30)	email character varying (40)	travel_id integer	comp_id integer	place_id integer
1	1	Bob Slam	976453276	bob@gmail.com	2	1	3
2	2	Maria Lonsey	637562834	lonsi_sas@gmail.com	3	2	5
3	3	Ilon Conra	964539821	g123kh@gmail.com	4	1	2
4	4	Rian Valdoro	638354286	samsobi@gmail.com	4	2	2
5	5	Lina Plaso	976245923	linopla@gmail.com	1	3	6
6	6	Tot Otot	675463865	totttot@gmail.com	1	2	3
7	7	OJSW	975553649	K64FK	2	2	2

Зробимо для 100000 полів:

Tables:

```
1.Customer
2.Trav_comp
3.Travel
4.Place
5.Back to menu
Choose table: 1
```

Adding random customer:

```
Enter the number: 100000
```



	customer_id [PK] integer	name character varying (50)	phone character varying (30)	email character varying (40)	travel_id integer	comp_id integer	place_id integer
1	1	Bob Slam	976453276	bob@gmail.com	2	1	3
2	2	Maria Lonsey	637562834	lonsi_sas@gmail.com	3	2	5
3	3	Ilon Conra	964539821	g123kh@gmail.com	4	1	2
4	4	Rian Valdoro	638354286	samsobi@gmail.com	4	2	2
5	5	Lina Plaso	976245923	linopla@gmail.com	1	3	6
6	6	Tot Otot	675463865	totttot@gmail.com	1	2	3
7	7	OJSW	975553649	K64FK	2	2	2
8	8	KQPP	732369843	H8KH	2	3	4
9	9	ACGE	747826667	I77KK	3	4	3
10	10	RKCO	236068775	Q31FT	2	2	1
11	11	MBBJ	584871528	S30WX	3	2	7
12	12	OPFF	915327493	H6XX	2	1	1
13	13	LSCY	546199955	U31QT	4	4	2
14	14	QAPI	952822663	S72UV	1	3	3
15	15	SCYC	324382424	J29SC	5	4	2
16	16	ENVE	855185617	G46JI	4	4	2
17	17	NRMA	111372672	G63US	2	3	7
18	18	YVPA	397798193	Q27YD	5	2	7
19	19	BIQE	547245268	G73JV	5	4	2
20	20	HLTU	226961700	A48RQ	2	1	4
21	21	GPOX	304497106	S90IY	4	3	4
22	22	VWCJ	620253907	Q30DP	2	3	7
23	23	FOTV	53385775	A8PS	4	2	7
24	24	GCGT	94158312	S34JO	4	2	3
25	25	RAFX	785490646	D94MD	3	3	1
26	26	BMGE	759496518	E18YE	5	1	6
27	27	PYBQ	561603984	Q24XC	2	4	3

28	28	FJOI	827453244	D45LM	3	3	7
29	29	GTYY	669208708	H71AO	1	1	4
30	30	MEXD	322183554	G77FN	3	1	3
31	31	VMOG	343117498	Y51YD	5	2	1
32	32	CYBS	67780576	R33UT	1	2	1
33	33	DDPC	555590735	P33QY	3	2	7
34	34	MBXW	528052425	A60KH	4	1	1
35	35	HWRO	645162255	L32VP	1	1	5
36	36	GAQU	602443616	E53UH	1	2	7
37	37	XFFK	62448176	B19KU	4	3	4
38	38	FMVW	620011769	J45MG	1	2	7
39	39	JNKO	132295082	T17WX	1	3	1
40	40	OTQV	362490493	T67MM	3	2	5
41	41	IJKI	534491809	S92AQ	5	2	1
42	42	WELT	924041630	P9BG	5	2	3
43	43	FSAT	145757188	L96XU	3	2	3
44	44	MDYU	73749547	C5WK	1	1	4
45	45	MXUN	803143711	N39WK	4	4	2
46	46	LEDT	952890775	Q32IX	1	1	1
47	47	RCNK	296344682	O86OA	1	4	2
48	48	GCEO	467068567	P63AR	4	4	1
49	49	JVCE	100056842	E39GJ	1	1	6
50	50	NCHN	804581973	E12JV	4	3	3
51	51	AAQR	358637873	L59NO	5	2	5
49	49	JVCE	100056842	E39GJ	1	1	6
50	50	NCHN	804581973	E12JV	4	3	3
51	51	AAQR	358637873	L59NO	5	2	5
52	52	POMI	375184230	J49JQ	1	4	4
53	53	FTBY	743396991	O58LR	2	3	1
54	54	ILMV	921349122	W85BH	1	2	5
55	55	PV VX	994215015	T60GH	1	3	3
56	56	UROK	283180442	N90KH	2	2	5
57	57	IRHI	881482043	V14SG	5	3	6
58	58	OUYL	341719674	V0SC	2	4	1
59	59	FALQ	885523147	A72QQ	4	1	7
60	60	JUXA	190406881	N21VK	4	3	6
61	61	MDTO	74129368	O97EE	2	2	1
62	62	YDNY	926857583	A90NE	5	3	5
63	63	PVUJ	31035978	N12BW	1	4	3
64	64	VBKY	746741373	R8DY	5	4	3
65	65	FDNX	261573908	L37UF	4	1	7
66	66	MSFT	374829574	H91JR	4	1	1
67	67	UATM	346096012	R38VX	3	4	7
68	68	CKBG	455677733	P57HX	1	1	3
69	69	CUUW	812587914	I82BC	3	1	4
70	70	PBBY	299623212	H11NL	2	2	3
71	71	HKTG	533988632	H75AC	5	4	7

71	71	HKTG	533988632	H75AC	5	4	7
72	72	DABK	244336721	J15LM	3	1	4
73	73	GOJB	842408865	E1RY	2	3	3
74	74	BFOT	15167416	H64QB	4	1	6
75	75	HKUC	678840740	X3CM	1	4	4
76	76	HPQU	977326851	K17VG	3	1	1
77	77	KOKW	786609861	I47KW	3	1	2
78	78	QJXH	805085726	Q49HV	5	1	4
79	79	PRIK	735462349	B94TX	1	2	7
80	80	ORBA	435820274	V97CS	4	1	5
81	81	QQKR	300924050	D110C	3	2	2
82	82	WPVV	227606342	L27SD	3	2	2
83	83	YDJW	988971783	M90UN	5	4	2
84	84	NRPR	223722667	C64LI	5	3	1
85	85	FAIG	71514617	J91NQ	2	3	2
86	86	KVCX	142789121	E94CX	5	1	4
87	87	HCYG	380005789	K37RE	5	4	5
88	88	EGSW	918848853	K87MD	5	1	6
89	89	HWKA	459711735	U11JM	4	1	2
90	90	OMEA	801989556	T83HU	3	2	5
91	91	HUHP	176681088	V69NI	2	3	5

**Для реалізації генерування «рандомізованих» даних був використаний такий SQL-запит:**

```
INSERT INTO customer (name, phone, email, travel_id, comp_id, place_id)
SELECT
chr(trunc(65 + random() * 25)::int) || chr(trunc(65 + random() * 25)::int) ||
chr(trunc(65 + random() * 25)::int) || chr(trunc(65 + random() * 25)::int),
trunc(random() * 1000000000),
chr(trunc(65 + random() * 25)::int) || trunc(random() * 100) || chr(trunc(65 +
random() * 25)::int) || chr(trunc(65 + random() * 25)::int),
trunc(random() * 5) + 1,
trunc(random() * 4) + 1,
trunc(random() * 7) + 1
FROM generate_series(1, %s)
```

Цей запит вставляє випадкові значення, в поля name, phone, email, travel\_id, comp\_id, place\_id.

generate\_series використовується для передачі кількості полів даних.

Пояснення функцій:

- chr(trunc(65 + random() \* 25)::int) || chr(trunc(65 + random() \* 25)::int) || chr(trunc(65 + random() \* 25)::int) || chr(trunc(65 + random() \* 25)::int) – вводить рандомні 4 літери латинського алфавіту

- `trunc(random() * 10000000000)` – генерує рандомні числа в діапазоні до 10000000000
- `chr(trunc(65 + random() * 25)::int) || trunc(random() * 100) || chr(trunc(65 + random() * 25)::int) || chr(trunc(65 + random() * 25)::int)` – генерує букву потім число до 100 а потім ще 2 букви
- `trunc(random() * 5) + 1` – генерує значення від 1 до 5 (`travel_id`)
- `trunc(random() * 4) + 1` – генерує значення від 1 до 4 (`comp_id`)
- `trunc(random() * 7) + 1` – генерує значення від 1 до 7 (`place_id`)

## Частина №3

Пошук даних:

```
Search:
1. Customer & Company
2. Travel type
3. Customer travel to list
4. Menu
Choose: |
```

Має 3 основні функції:

### 1. Customer & Company

При виборі цієї опції таблиці будуть відфільтровані так, що утворена таблиця буде містити інформацію про `comp_id` (і буде відсортована по цьому значенню), ім'я покупця та компанію яка здійснює транспортування:

```
TRcomp ID: 1, Customer name: Ilon Conra, Company name:Altion
TRcomp ID: 1, Customer name: HPQU, Company name:Altion
TRcomp ID: 1, Customer name: BMGE, Company name:Altion
TRcomp ID: 1, Customer name: BFOT, Company name:Altion
TRcomp ID: 1, Customer name: DABK, Company name:Altion
TRcomp ID: 1, Customer name: GTYV, Company name:Altion
TRcomp ID: 1, Customer name: MEXD, Company name:Altion
TRcomp ID: 2, Customer name: EVBD, Company name:Eurostar
TRcomp ID: 2, Customer name: Maria Lonsey, Company name:Eurostar
TRcomp ID: 2, Customer name: Rian Valdoro, Company name:Eurostar
TRcomp ID: 2, Customer name: Tot Otot, Company name:Eurostar
TRcomp ID: 2, Customer name: OJSW, Company name:Eurostar
TRcomp ID: 2, Customer name: RKCO, Company name:Eurostar
TRcomp ID: 2, Customer name: MBBJ, Company name:Eurostar
```

Так як в таблиці customer 100 тис полів, зробимо ліміт 100, отримаємо:

```
TRcomp ID: 4, Customer name: SSEB, Company name:Lambada  
TRcomp ID: 4, Customer name: LSCY, Company name:Lambada  
Time: 7.00 мс
```

Для реалізації був використаний наступний SQL-запит:

```
SELECT  
customer.comp_id,  
customer.name AS customer_name,  
trcomp.comp_name AS trcomp_name  
FROM  
customer  
LEFT JOIN  
trcomp ON trcomp.comp_id = customer.comp_id  
ORDER BY  
comp_id;
```

Цей запит використовує оператор LEFT JOIN, щоб об'єднати таблиці по comp\_id, і впорядковує по значенню comp\_id. Відповідно поля які виводяться вказані в SELECT.

## 2. Travel type

При виборі цієї опції, буде створена таблиця в якій буде вказані пасажир(с)и(customer.name) і їхні рейси(travel.id), але при цьому, потрібно буде ввести номер рейсу з клавіатури і буде показано всіх пасажирів зареєстрованих на цей рейс:

```
You need to enter the travel_id.  
Enter the number: 2  
  
Sorting by travel_id:  
customer: Bob Slam, travel_id: 2  
customer: OJSW, travel_id: 2  
customer: KQPP, travel_id: 2  
customer: RKCO, travel_id: 2  
customer: OPFF, travel_id: 2  
customer: NRMA, travel_id: 2  
customer: HLTU, travel_id: 2  
customer: VWCJ, travel_id: 2  
customer: PYBQ, travel_id: 2  
customer: FTBY, travel_id: 2  
customer: UROK, travel_id: 2
```

Відповідно теж використаємо обмеження на 100 полів, отримаємо час:

```
customer: ISIH, travel_id: 2  
customer: EVBD, travel_id: 2  
Time: 3787.66 мс
```

Для реалізації був використаний наступний SQL-запит:

```
SELECT  
customer.name AS customer_name,  
customer.travel_id  
FROM  
customer  
WHERE  
customer.travel_id = %s;
```

Виводиться ім'я пасажирів і id поїздки, оператор WHERE фільтрує id поїздки відповідно до значення введеного з клавіатури.

### 3. Customer travel to list

При виборі цієї опції будується таблиця в якій знаходяться ідентифікатори місць поїздки, ім'я зареєстрованого пасажирів і місто куди відбувається поїздка, вся таблиця впорядкована відповідно до ідентифікатору міста поїздки.

```
Travel id: 1 Name customer: NRPR Travel to: Lviv.  
Travel id: 1 Name customer: OPFF Travel to: Lviv.  
Travel id: 1 Name customer: OUYL Travel to: Lviv.  
Travel id: 1 Name customer: RAFX Travel to: Lviv.  
Travel id: 1 Name customer: RKCO Travel to: Lviv.  
Travel id: 1 Name customer: VMOG Travel to: Lviv.  
Travel id: 2 Name customer: BIQE Travel to: Krakow.  
Travel id: 2 Name customer: ENVE Travel to: Krakow.  
Travel id: 2 Name customer: EVBD Travel to: Krakow.  
Travel id: 2 Name customer: FAIG Travel to: Krakow.  
Travel id: 2 Name customer: FTYF Travel to: Krakow.  
Travel id: 2 Name customer: HWKA Travel to: Krakow.
```

```
Travel id: 2 Name customer: QQKR Travel to: Krakow.  
Travel id: 2 Name customer: RCNK Travel to: Krakow.  
Travel id: 2 Name customer: Rian Valdoro Travel to: Krakow.  
Travel id: 2 Name customer: SCYC Travel to: Krakow.  
Travel id: 2 Name customer: WPVV Travel to: Krakow.  
Travel id: 2 Name customer: XKQR Travel to: Krakow.  
Travel id: 2 Name customer: YDJW Travel to: Krakow.  
Travel id: 3 Name customer: ACGE Travel to: Rome.  
Travel id: 3 Name customer: Bob Slam Travel to: Rome.  
Travel id: 3 Name customer: CKBG Travel to: Rome.  
Travel id: 3 Name customer: FSAT Travel to: Rome.  
Travel id: 3 Name customer: GCGT Travel to: Rome.  
Travel id: 3 Name customer: GOJB Travel to: Rome.  
Travel id: 3 Name customer: ISIH Travel to: Rome.
```

Відповідно теж використаємо обмеження на 100 полів, отримаємо час:

```
Travel id: 7 Name customer: VWCJ Travel to: Riga.  
Travel id: 7 Name customer: YVPA Travel to: Riga.  
Time: 6.01 mc
```

Для реалізації був використаний наступний SQL-запит:

```
SELECT  
customer.place_id,  
customer.name AS customer_name,  
place.city AS place_city  
FROM  
customer  
LEFT JOIN  
place ON place.place_id = customer.place_id  
GROUP BY  
customer.place_id, place.place_id, customer.name  
ORDER BY  
place_id;
```

В таблицю виводяться значення з SELECT, при цьому синхронізують значення по полю place.id, групуються за ідентифікатором місця і назвою, і впорядковуються відповідно place\_id.

## Частина №4

MVC розшифровується як "модель-подання-контролер" (від англ. modelview-controller). Це спосіб організації коду, який передбачає виділення блоків, що відповідають за рішення різних завдань. Один блок відповідає за дані програми, інший відповідає за зовнішній вигляд, а третій контролює роботу програми.

Компоненти MVC:

1. Модель - цей компонент відповідає за дані, а також визначає структуру додатка.
  2. Представлення - цей компонент відповідає за взаємодію з користувачем.
  3. Контролер - цей компонент відповідає за зв'язок між "bdcon" та "visual".
- По суті, це мозок MVC-дodatка.

main.py –початок програми, запускає початковий інтерфейс.

dbcon.py – виконує дії з БД.

visual.py – файл, що відповідає за візуальну частину програми і взаємодію з користувачем.

progbody.py – виконує підключення до бази даних, обробляє введені користувачем дані, використовує функції dbcon.py, тобто є мозком програми.

## Код програми

### main.py

```
from progbody import Controller

if __name__ == "__main__":
    controller = Controller()
    controller.run()
```

### dbcon.py

```
import psycopg2

class Model:
    def __init__(self):
        self.conn = psycopg2.connect(
            dbname='postgres',
            user='postgres',
            password='123qwertyuiop',
            host='localhost',
            port=5432
        )

    def add_customer(self, name, phone, email, travel_id, comp_id, place_id):
        c = self.conn.cursor()
```



```

        c.execute('INSERT INTO customer (name, phone, email, travel_id,
comp_id, place_id) VALUES (%s, %s, %s, %s, %s, %s)', (name, phone, email,
travel_id, comp_id, place_id))
        self.conn.commit()

    def add_trcomp(self, comp_name, comp_address, comp_phone):
        c = self.conn.cursor()
        c.execute('INSERT INTO trcomp (comp_name, comp_address,
comp_phone)VALUES (%s, %s, %s)', (comp_name, comp_address, comp_phone))
        self.conn.commit()

    def add_travel(self, time, form, pay):
        c = self.conn.cursor()
        c.execute('INSERT INTO travel (time, form, pay) VALUES (%s, %s, %s)',
(time, form, pay))
        self.conn.commit()

    def add_place(self, country, city):
        c = self.conn.cursor()
        c.execute('INSERT INTO place (country, city) VALUES (%s, %s)',
(country, city))
        self.conn.commit()

    def get_all_customers(self):
        c = self.conn.cursor()
        c.execute('SELECT * FROM customer')
        return c.fetchall()

    def get_all_trcomps(self):
        c = self.conn.cursor()
        c.execute('SELECT * FROM trcomp')
        return c.fetchall()

    def get_all_travels(self):
        c = self.conn.cursor()
        c.execute('SELECT * FROM travel')
        return c.fetchall()

    def get_all_places(self):
        c = self.conn.cursor()
        c.execute('SELECT * FROM place')
        return c.fetchall()

    def update_customer(self, customer_id, name, phone, email, travel_id,
comp_id, place_id):
        c = self.conn.cursor()
        c.execute('UPDATE customer SET name=%s, phone=%s, email=%s,
travel_id=%s, comp_id=%s, place_id=%s WHERE customer_id=%s', (name, phone,
email, travel_id, comp_id, place_id, customer_id))
        self.conn.commit()

    def update_trcomp(self, comp_id, comp_name, comp_address, comp_phone):
        c = self.conn.cursor()
        c.execute('UPDATE trcomp SET comp_name=%s, comp_address=%s,
comp_phone=%s WHERE comp_id=%s', (comp_name, comp_address, comp_phone,

```

```

comp_id))
    self.conn.commit()

    def update_travel(self, travel_id, time, form, pay):
        c = self.conn.cursor()
        c.execute('UPDATE travel SET time=%s, form=%s, pay=%s WHERE
travel_id=%s', (time, form, pay, travel_id))
        self.conn.commit()

    def update_place(self, place_id, country, city):
        c = self.conn.cursor()
        c.execute('UPDATE place SET country=%s, city=%s WHERE place_id=%s',
(country, city, place_id))
        self.conn.commit()

    def delete_customer(self, customer_id):
        c = self.conn.cursor()
        c.execute('DELETE FROM customer WHERE customer_id=%s',
(customer_id,))
        self.conn.commit()

    def delete_trcomp(self, trcomp_id):
        c = self.conn.cursor()
        c.execute('DELETE FROM trcomp WHERE trcomp_id=%s', (trcomp_id,))
        self.conn.commit()

    def delete_travel(self, travel_id):
        c = self.conn.cursor()
        c.execute('DELETE FROM travel WHERE travel_id=%s', (travel_id,))
        self.conn.commit()

    def delete_place(self, place_id):
        c = self.conn.cursor()
        c.execute('DELETE FROM place WHERE place_id=%s', (place_id,))
        self.conn.commit()

    def add_random_fields(self, number):
        c = self.conn.cursor()
        c.execute(
            'INSERT INTO customer (name, phone, email, travel_id, comp_id,
place_id) SELECT chr(trunc(65 + random() * 25)::int) || chr(trunc(65 +
random() * 25)::int) || chr(trunc(65 + random() * 25)::int) || chr(trunc(65 +
random() * 25)::int), trunc(random() * 10000000000), chr(trunc(65 + random() *
25)::int) || trunc(random() * 100) || chr(trunc(65 + random() * 25)::int) ||
chr(trunc(65 + random() * 25)::int), trunc(random() * 5) + 1, trunc(random()
* 4) + 1, trunc(random() * 7) + 1 FROM generate_series(1, %s)', (number,))
        self.conn.commit()

    def Customer_Company12(self):
        c = self.conn.cursor()
        c.execute('SELECT customer.comp_id, customer.name AS customer_name,
trcomp.comp_name AS trcomp_name FROM customer LEFT JOIN trcomp ON
trcomp.comp_id = customer.comp_id ORDER BY comp_id;')
        return c.fetchall()

    def Travel_Type12(self, number):

```

```

        c = self.conn.cursor()
        c.execute('SELECT customer.name AS customer_name, customer.travel_id
FROM customer WHERE customer.travel_id = %s;', (number,))
        return c.fetchall()

    def Customer_Travel_To12(self):
        c = self.conn.cursor()
        c.execute('SELECT customer.place_id, customer.name AS customer_name,
place.city AS place_city FROM customer LEFT JOIN place ON place.place_id =
customer.place_id GROUP BY customer.place_id, place.place_id, customer.name
ORDER BY place_id;')
        return c.fetchall()

```

## visual.py

```
class View:
```

```

    def show_menu(self):

        self.show_message("\nMenu:")
        self.show_message("1.Add line")
        self.show_message('2.Add random line(for "customers"')
        self.show_message("3.Show table")
        self.show_message("4.Update line")
        self.show_message("5.Delete line")
        self.show_message("6.Search")
        self.show_message("7.Exit")
        choice = input("Choose: ")
        return choice

    def show_tables(self):
        self.show_message("\nTables:")
        self.show_message("1.Customer")
        self.show_message("2.Trav_comp")
        self.show_message("3.Travel")
        self.show_message("4.Place")
        self.show_message("5.Back to menu")
        table = input("Choose table: ")
        return table

    def show_search(self):
        self.show_message("\nSearch:")
        self.show_message("1. Customer & Company")
        self.show_message("2. Travel type")
        self.show_message("3. Customer travel to list")
        self.show_message("4. Menu")
        choice = input("Choose: ")
        return choice

    def show_travels(self, travels):
        print("\nTravels:")
        for travel in travels:
            print(f"ID: {travel[0]}, Time: {travel[1]}, Form: {travel[2]},
Pay: {travel[3]}")

    def show_places(self, places):
        print("\nplaces:")
        for place in places:

```

```

        print(f"ID: {place[0]}, Country: {place[1]}, City: {place[2]}")

    def show_trcomps(self, trcomps):
        print("\ntrcomps:")
        for trcomp in trcomps:
            print(f"ID: {trcomp[0]}, Name: {trcomp[1]}, Address: {trcomp[2]},
Phone: {trcomp[3]}")

    def show_customers(self, customers):
        print("\ncustomer:")
        for customer in customers:
            print(
                f"ID: {customer[0]}, Name: {customer[1]}, Phone:
{customer[2]}, Email: {customer[3]}, travel_id: {customer[4]}, comp_id:
{customer[5]}, place_id: {customer[6]}")

    def Customer_Company12_Show(self, rows):
        print("\nSort by Trcomp_id:")
        for row in rows:
            print(f"TRcomp ID: {row[0]}, Customer name: {row[1]}, Company
name:{row[2]}")

    def Travel_Type12_Show(self, rows):
        print("\nSorting by travel_id:")
        for row in rows:
            print(f"customer: {row[0]}, travel_id: {row[1]}")

    def Customer_Travel_Tol2_Show(self, rows):
        print("\nTravel-Name-Place:")
        for row in rows:
            print(f'Travel id: {row[0]} Name customer: {row[1]} Travel to:
{row[2]}'.)

    def get_customer_input(self):
        while True:
            try:
                name = input("Enter customer name: ")
                if name.strip():
                    break
            else:
                print("Name cannot be empty.")
        except ValueError:
            print("It must be a string.")
        while True:
            try:
                phone = input("Enter customer phone: ")
                if phone.strip():
                    phone = float(phone)
                    break
            else:
                print("Phone cannot be empty.")
        except ValueError:
            print("It must be a number.")
        while True:
            try:
                email = input("Enter customer email: ")
                if email.strip():
                    break
            else:
                print("Email cannot be empty.")
        except ValueError:
            print("It must be a string.")
        while True:

```

```

        try:
            travel_id = int(input("Enter customer travel_id: "))
            break
        except ValueError:
            print("Travel_id must be a number.")
    while True:
        try:
            comp_id = int(input("Enter customer comp_id: "))
            break
        except ValueError:
            print("Comp_id must be a number.")
    while True:
        try:
            place_id = int(input("Enter customer place_id: "))
            break
        except ValueError:
            print("Place_id must be a number.")
    return name, phone, email, travel_id, comp_id, place_id

def get_trcomp_input(self):
    while True:
        try:
            comp_name = input("Enter trcomp comp_name: ")
            if comp_name.strip():
                break
            else:
                print("Comp_name cannot be empty.")
        except ValueError:
            print("It must be a string.")
    while True:
        try:
            comp_address = input("Enter trcomp comp_address: ")
            if comp_address.strip():
                break
            else:
                print("Comp_address cannot be empty.")
        except ValueError:
            print("It must be a string.")
    while True:
        try:
            comp_phone = input("Enter trcomp comp_phone: ")
            if comp_phone.strip():
                break
            else:
                print("Comp_phone cannot be empty.")
        except ValueError:
            print("It must be a string.")
    return comp_name, comp_address, comp_phone

def get_travel_input(self):
    while True:
        try:
            time = input("Enter travel time: ")
            if time.strip():
                break
            else:
                print("Time cannot be empty.")
        except ValueError:
            print("It must be a string.")
    while True:
        try:
            form = input("Enter travel form: ")

```

```

        if form.strip():
            break
        else:
            print("Form cannot be empty.")
    except ValueError:
        print("It must be a string.")
while True:
    try:
        pay = input("Enter travel pay: ")
        if pay.strip():
            break
        else:
            print("Pay cannot be empty.")
    except ValueError:
        print("It must be a string.")
return time, form, pay

def get_place_input(self):
    while True:
        try:
            country = input("Enter place country: ")
            if country.strip():
                break
            else:
                print("Country cannot be empty.")
        except ValueError:
            print("It must be a string.")
    while True:
        try:
            city = input("Enter place city: ")
            if city.strip():
                break
            else:
                print("City cannot be empty.")
        except ValueError:
            print("It must be a string.")
    return country, city

def get_customer_id(self):
    while True:
        try:
            id = int(input("Enter customer ID: "))
            break
        except ValueError:
            print("It must be a number.")
    return id

def get_trcomp_id(self):
    while True:
        try:
            id = int(input("Enter trcomp ID: "))
            break
        except ValueError:
            print("It must be a number.")
    return id

def get_travel_id(self):
    while True:
        try:
            id = int(input("Enter travle ID: "))

```

```

        break
    except ValueError:
        print("It must be a number.")
    return id

def get_place_id(self):
    while True:
        try:
            id = int(input("Enter place ID: "))
            break
        except ValueError:
            print("It must be a number.")
    return id

def get_task_id(self):
    while True:
        try:
            id = int(input("Enter task ID: "))
            break
        except ValueError:
            print("It must be a number.")
    return id

def show_message(self, message):
    print(message)

def get_number(self):
    while True:
        try:
            number = int(input("Enter the number: "))
            break
        except ValueError:
            print("It must be a number.")
    return number

```

## progbody.py

```

import time
from dbcon import Model
from visual import View

class Controller:
    def __init__(self):
        self.model = Model()
        self.view = View()

    def run(self):
        while True:
            choice = self.view.show_menu()
            if choice == '7':
                break
            if choice == '6':
                self.process_search_option()
            elif choice in ['1', '2', '3', '4', '5']:
                self.process_menu_choice(choice)
            else:
                self.view.show_message("Wrong choice. Try again.")

```

```

def process_menu_choice(self, choice):
    while True:
        table = self.view.show_tables()
        if table == '6':
            break
        if choice == '1':
            self.process_add_option(table)
        elif choice == '2':
            self.process_add_random_option(table)
        elif choice == '3':
            self.process_view_option(table)
        elif choice == '4':
            self.process_update_option(table)
        elif choice == '5':
            self.process_delete_option(table)

def process_add_option(self, table):
    if table == '1':
        self.view.show_message("\nAdding customer:")
        self.add_customer()
    elif table == '2':
        self.view.show_message("\nAdding trcomp:")
        self.add_trcomp()
    elif table == '3':
        self.view.show_message("\nAdding travel:")
        self.add_travel()
    elif table == '4':
        self.view.show_message("\nAdding place:")
        self.add_place()
    else:
        self.view.show_message("Wrong choice. Try again.")

def process_add_random_option(self, table):
    if table == '1':
        self.view.show_message("\nAdding random customer:")
        self.add_random_fields()
    else:
        self.view.show_message("Wrong choice. Try again.")

def process_view_option(self, table):
    if table == '1':
        self.view_customers()
    elif table == '2':
        self.view_trcomps()
    elif table == '3':
        self.view_travels()
    elif table == '4':
        self.view_places()
    elif table == '5':
        self.view.show_menu()
    else:
        self.view.show_message("Wrong choice. Try again.")

def process_update_option(self, table):
    if table == '1':
        self.view.show_message("\nUpdating customer:")
        self.update_customer()
    elif table == '2':
        self.view.show_message("\nUpdating trcomp:")

```



```

        self.update_trcomp()
    elif table == '3':
        self.view.show_message("\nUpdating travel:")
        self.update_travel()
    elif table == '4':
        self.view.show_message("\nUpdating place:")
        self.update_place()
    else:
        self.view.show_message("Wrong choice. Try again.")

def process_delete_option(self, table):
    if table == '1':
        self.view.show_message("\nDeleting customer:")
        self.delete_customer()
    elif table == '2':
        self.view.show_message("\nDeleting trcomp:")
        self.delete_trcomp()
    elif table == '3':
        self.view.show_message("\nDeleting travel:")
        self.delete_travel()
    elif table == '4':
        self.view.show_message("\nDeleting place:")
        self.delete_place()
    else:
        self.view.show_message("Wrong choice. Try again.")

def process_search_option(self):
    option = self.view.show_search()
    if option == '1':
        start_time = time.time()
        self.Customer_Company12_Show()
        end_time = time.time()
        elapsed_time = (end_time - start_time) * 1000
        print(f"Time: {elapsed_time:.2f} mc")
    elif option == '2':
        start_time = time.time()
        self.Travel_Type12_Show()
        end_time = time.time()
        elapsed_time = (end_time - start_time) * 1000
        print(f"Time: {elapsed_time:.2f} mc")
    elif option == '3':
        start_time = time.time()
        self.Customer_Travel_To12_Show()
        end_time = time.time()
        elapsed_time = (end_time - start_time) * 1000
        print(f"Time: {elapsed_time:.2f} mc")
    else:
        self.view.show_menu()

def add_customer(self):
    try:
        name, phone, email, travel_id, comp_id, place_id =
self.view.get_customer_input()
        self.model.add_customer(name, phone, email, travel_id, comp_id,
place_id)
        self.view.show_message("Customer added successfully!")
    except Exception as e:
        self.view.show_message(f"Something went wrong: {e}")

def add_trcomp(self):
    try:
        comp_name, comp_address, comp_phone =

```

```

self.view.get_trcomp_input()
    self.model.add_trcomp(comp_name, comp_address, comp_phone)
    self.view.show_message("Trcomp added successfully!")
except Exception as e:
    self.view.show_message(f"Something went wrong: {e}")

def add_travel(self):
    try:
        time, form, pay = self.view.get_travel_input()
        self.model.add_travel(time, form, pay)
        self.view.show_message("Travel added successfully!")
    except Exception as e:
        self.view.show_message(f"Something went wrong: {e}")

def add_place(self):
    try:
        country, city = self.view.get_place_input()
        self.model.add_place(country, city)
        self.view.show_message("Place added successfully!")
    except Exception as e:
        self.view.show_message(f"Something went wrong: {e}")

def view_customers(self):
    try:
        customers = self.model.get_all_customers()
        self.view.show_customers(customers)
    except Exception as e:
        self.view.show_message(f"Something went wrong: {e}")

def view_trcomps(self):
    try:
        trcomps = self.model.get_all_trcomps()
        self.view.show_trcomps(trcomps)
    except Exception as e:
        self.view.show_message(f"Something went wrong: {e}")

def view_travels(self):
    try:
        travels = self.model.get_all_travels()
        self.view.show_travels(travels)
    except Exception as e:
        self.view.show_message(f"Something went wrong: {e}")

def view_places(self):
    try:
        places = self.model.get_all_places()
        self.view.show_places(places)
    except Exception as e:
        self.view.show_message(f"Something went wrong: {e}")

def Customer_Company12_Show(self):
    try:
        rows = self.model.Customer_Company12()
        self.view.Customer_Company12_Show(rows)
    except Exception as e:
        self.view.show_message(f"Something went wrong: {e}")

def Travel_Type12_Show(self):

```

```

        try:
            self.view.show_message("\nYou need to enter the travel_id.")
            number = self.view.get_number()
            rows = self.model.Travel_Type12(number)
            self.view.Travel_Type12_Show(rows)
        except Exception as e:
            self.view.show_message(f"Something went wrong: {e}")

    def Customer_Travel_To12_Show(self):
        try:
            rows = self.model.Customer_Travel_To12()
            self.view.Customer_Travel_To12_Show(rows)
        except Exception as e:
            self.view.show_message(f"Something went wrong: {e}")

    def update_customer(self):
        try:
            customer_id = self.view.get_customer_id()
            name, phone, email, travel_id, comp_id, place_id =
self.view.get_customer_input()
            self.model.update_customer(customer_id, name, phone, email,
travel_id, comp_id, place_id)
            self.view.show_message("Customer updated successfully!")
        except Exception as e:
            self.view.show_message(f"Something went wrong: {e}")

    def update_trcomp(self):
        try:
            comp_id = self.view.get_trcomp_id()
            comp_name, comp_address, comp_phone =
self.view.get_trcomp_input()
            self.model.update_trcomp(comp_id, comp_name, comp_address,
comp_phone)
            self.view.show_message("Trcomp updated successfully!")
        except Exception as e:
            self.view.show_message(f"Something went wrong: {e}")

    def update_travel(self):
        try:
            travel_id = self.view.get_travel_id()
            time, form, pay = self.view.get_travel_input()
            self.model.update_travel(travel_id, time, form, pay)
            self.view.show_message("Travel updated successfully!")
        except Exception as e:
            self.view.show_message(f"Something went wrong: {e}")

    def update_place(self):
        try:
            place_id = self.view.get_place_id()
            country, city = self.view.get_place_input()
            self.model.update_place(place_id, country, city)
            self.view.show_message("Place updated successfully!")
        except Exception as e:
            self.view.show_message(f"Something went wrong: {e}")

    def delete_customer(self):
        try:
            customer_id = self.view.get_customer_id()
            self.model.delete_customer(customer_id)
            self.view.show_message("Customer deleted successfully!")

```

```
except Exception as e:
    self.view.show_message(f"Something went wrong: {e}")

def delete_trcomp(self):
    try:
        trcomp_id = self.view.get_trcomp_id()
        self.model.delete_trcomp(trcomp_id)
        self.view.show_message("Trcomp deleted successfully!")
    except Exception as e:
        self.view.show_message(f"Something went wrong: {e}")

def delete_travel(self):
    try:
        travel_id = self.view.get_travel_id()
        self.model.delete_travel(travel_id)
        self.view.show_message("Travel deleted successfully!")
    except Exception as e:
        self.view.show_message(f"Something went wrong: {e}")

def delete_place(self):
    try:
        place_id = self.view.get_place_id()
        self.model.delete_place(place_id)
        self.view.show_message("Place deleted successfully!")
    except Exception as e:
        self.view.show_message(f"Something went wrong: {e}")

def add_random_fields(self):
    try:
        number = self.view.get_number()
        self.model.add_random_fields(number)
        self.view.show_message("Random fields added successfully!")
    except Exception as e:
        self.view.show_message(f"Something went wrong: {e}")
```