1. Создать таблицу

```
CREATE TABLE user_actions
(
    user_id UInt64,
    action String,
    expense UInt64
)
ENGINE = MergeTree()
ORDER BY user_id;
```

```
CREATE TABLE user_actions
(
    user_id UInt64,
    action String,
    expense UInt64
)
ENGINE = MergeTree()
ORDER BY user_id;
```

Statistics 1

Name	Value
Updated Rows	0
Query	CREATE TABLE user_actions
	(
	user_id UInt64,
	action String,
	expense UInt64
	)
	ENGINE = MergeTree()
	ORDER BY user_id
Start time	Fri Nov 08 07:24:51 MSK 2024
Finish time	Fri Nov 08 07:24:51 MSK 2024

2. Создать словарь, в качестве ключа user_id, в качестве атрибута email String, источник словаря – этот же экземпляр ClickHouse

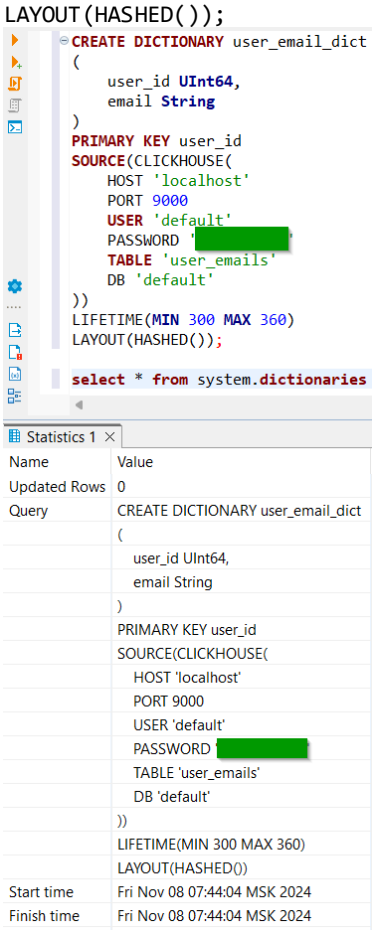
```
CREATE TABLE user_emails
(
    user_id UInt64,
    email String
)
ENGINE = MergeTree()
ORDER BY user_id;
```

```
CREATE TABLE user_emails
(
    user_id UInt64,
    email String
)
ENGINE = MergeTree()
ORDER BY user_id;
```

Statistics 1

Name	Value
Updated Rows	0
Query	CREATE TABLE user_emails
	(
	user_id UInt64,
	email String
	)
	ENGINE = MergeTree()
	ORDER BY user_id
Start time	Fri Nov 08 07:28:42 MSK 2024
Finish time	Fri Nov 08 07:28:42 MSK 2024

```
CREATE DICTIONARY user_email_dict
(
    user_id UInt64,
    email String
)
PRIMARY KEY user_id
SOURCE(CLICKHOUSE(
    HOST 'localhost'
    PORT 9000
    USER 'default'
    PASSWORD '*****'
    TABLE 'user_emails'
    DB 'default'
))
LIFETIME(MIN 300 MAX 360)
LAYOUT(HASHED());
```



3. Наполнить таблицу и источник любыми данными, с низкоардинальными значениями для поля action и хотя бы по несколько повторящихся строк для каждого user_id

```
INSERT INTO user_actions (user_id, action, expense)
VALUES
    (1, 'login', 0),
    (1, 'view_product', 0),
    (1, 'add_to_cart', 0),
    (1, 'purchase', 100),
    (2, 'login', 0),
    (2, 'view_product', 0),
    (2, 'add_to_cart', 0),
    (2, 'purchase', 150),
    (3, 'login', 0),
    (3, 'view_product', 0),
    (3, 'add_to_cart', 0),
    (3, 'view_product', 0),
    (3, 'purchase', 200),
    (4, 'login', 0),
    (4, 'view_product', 0),
    (4, 'add_to_cart', 0),
```

```
    (4, 'remove_from_cart', 0),
    (4, 'logout', 0),
    (5, 'login', 0),
    (5, 'view_product', 0),
    (5, 'add_to_cart', 0),
    (5, 'purchase', 75);
```

```sql
INSERT INTO user_actions (user_id, action, expense)
VALUES
    (1, 'login', 0),
    (1, 'view_product', 0),
    (1, 'add_to_cart', 0),
    (1, 'purchase', 100),
    (2, 'login', 0),
    (2, 'view_product', 0),
    (2, 'add_to_cart', 0),
    (2, 'purchase', 150),
    (3, 'login', 0),
    (3, 'view_product', 0),
    (3, 'add_to_cart', 0),
    (3, 'view_product', 0),
    (3, 'purchase', 200),
    (4, 'login', 0),
    (4, 'view_product', 0),
    (4, 'add_to_cart', 0),
    (4, 'remove_from_cart', 0),
    (4, 'logout', 0),
    (5, 'login', 0),
    (5, 'view_product', 0),
    (5, 'add_to_cart', 0),
    (5, 'purchase', 75);
```

Statistics 1

Name	Value
Updated Rows	22
Query	INSERT INTO user_actions (user_id, action, expense)
	VALUES

```sql
INSERT INTO user_emails (user_id, email)
VALUES
    (1, 'user1@example.com'),
    (2, 'user2@example.com'),
    (3, 'user3@example.com'),
    (4, 'user4@example.com'),
    (5, 'user5@example.com');
```

```sql
INSERT INTO user_emails (user_id, email)
VALUES
    (1, 'user1@example.com'),
    (2, 'user2@example.com'),
    (3, 'user3@example.com'),
    (4, 'user4@example.com'),
    (5, 'user5@example.com');
```

Statistics 1

Name	Value
Updated Rows	5
Query	INSERT INTO user_emails (user_id, email)
	VALUES
	(1, 'user1@example.com'),
	(2, 'user2@example.com'),
	(3, 'user3@example.com'),
	(4, 'user4@example.com'),
	(5, 'user5@example.com')
Start time	Fri Nov 08 07:35:51 MSK 2024
Finish time	Fri Nov 08 07:35:51 MSK 2024

4. **<u>Написать SELECT, возвращающий:</u>**
   - **email при помощи dictGet,**
   - **аккамулятивную сумму expense, с окном по action**
   - **сортировка по email**

```
SELECT
    dictGet('user_email_dict', 'email', user_id) AS email,
    action,
    expense,
    sum(expense) OVER (PARTITION BY action ORDER BY user_id) AS cumulative_expense
FROM user_actions
ORDER BY email;
```

```
SELECT
    dictGet('user_email_dict', 'email', user_id) AS email,
    action,
    expense,
    sum(expense) OVER (PARTITION BY action ORDER BY user_id) AS cumulative_expense
FROM user_actions
ORDER BY email;
```

unknown 1

SELECT dictGet('user_email_dict', 'email', user_id) AS email, action, expense, s | Enter a SQL expression to filter results (us

Grid | Text

	email	action	expense	cumulative_expense
1	user1@example.com	add_to_cart	0	0
2	user1@example.com	login	0	0
3	user1@example.com	purchase	100	100
4	user1@example.com	view_product	0	0
5	user2@example.com	add_to_cart	0	0
6	user2@example.com	login	0	0
7	user2@example.com	purchase	150	250
8	user2@example.com	view_product	0	0
9	user3@example.com	add_to_cart	0	0
10	user3@example.com	login	0	0
11	user3@example.com	purchase	200	450
12	user3@example.com	view_product	0	0
13	user3@example.com	view_product	0	0
14	user4@example.com	add_to_cart	0	0
15	user4@example.com	login	0	0
16	user4@example.com	logout	0	0
17	user4@example.com	remove_from_cart	0	0
18	user4@example.com	view_product	0	0
19	user5@example.com	add_to_cart	0	0
20	user5@example.com	login	0	0
21	user5@example.com	purchase	75	525
22	user5@example.com	view_product	0	0