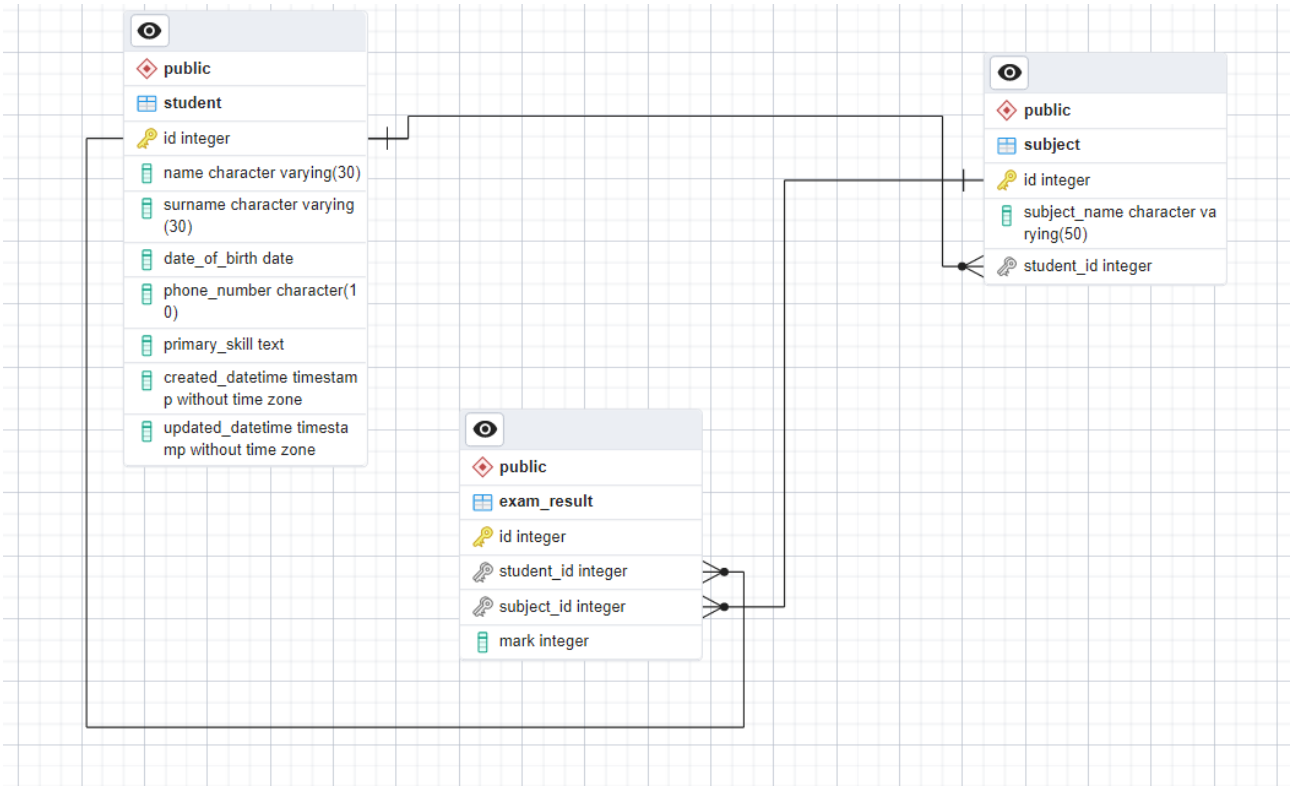
Task ‘Design database for CDP program.’

Your DB should store information about students (name, surname, date of birth, phone numbers, primary skill, created_datetime, updated_datetime etc.), subjects (subject name, tutor, etc.) and exam results (student, subject, mark).

Please add appropriate constraints (primary keys, foreign keys, indexes, etc.).

Design such kind of database for PostrgeSQL. Show your design in some suitable way (PDF, PNG, etc). (1 point)



Try different kind of indexes (B-tree, Hash, GIN, GIST) for your fields. Analyze performance for each of the indexes (use ANALYZE and EXPLAIN). Check the size of the index. Try to set index before inserting test data and after. What was the time? Test data:

a. 100K of users

b. 1K of subjects

c. 1 million of marks

Test queries:

a. Find user by name (exact match)

b. Find user by surname (partial match)

c. Find user by phone number (partial match)

d. Find user with marks by user surname (partial match)

Results before inserting test data:

a. Find user by name

```
1  -- 1.
2  -- CREATE INDEX idx_student_name ON student(name);
3  -- set enable_seqscan = off;
4  explain ANALYZE select * from student where name = 'Tom';
5
6  -- 2.
7  -- CREATE EXTENSION pg_trgm;
```

Data Output | Messages | Notifications

	QUERY PLAN text
1	Index Scan using idx_student_name on student (cost=0.13..8.14 rows=1 width=256) (actual time=1.332..1.334 rows=1 loop…
2	Index Cond: ((name)::text = 'Tom'::text)
3	Planning Time: 17.718 ms
4	Execution Time: 1.362 ms

Execution time using **B-tree index** on student_name field: **1.362 ms**.

```
1  -- 1.
2  -- CREATE INDEX idx_student_name ON student(name);
3  CREATE INDEX idx_hash_student_name ON student USING hash(name);
4  -- set enable_seqscan = off;
5  explain ANALYZE select * from student where name = 'Tom';
6
```

Data Output | Messages | Notifications

	QUERY PLAN text
1	Index Scan using idx_hash_student_name on student (cost=0.00..8.02 rows=1 width=256) (actual time=0.022..0.023 rows=1 loops=1)
2	Index Cond: ((name)::text = 'Tom'::text)
3	Planning Time: 1.345 ms
4	Execution Time: 0.032 ms

Execution time using **HASH index** on student_name field: **0.032 ms**.

b. Find user by surname

```
6  -- 2.
7  -- CREATE EXTENSION pg_trgm;
8  -- CREATE INDEX idx_student_surname_gist ON student USING gist (surname gist_trgm_ops);
9  -- SET enable_bitmapscan TO off;
10 explain ANALYZE select * from student where surname like 'Ku%';
11
```

Data Output | Messages | Notifications

	QUERY PLAN text
1	Index Scan using idx_student_surname_gist on student (cost=0.13..8.14 rows=1 width=256) (actual time=7.245..7.246 rows=1 loops=1)
2	Index Cond: ((surname)::text ~~ 'Ku%'::text)
3	Planning Time: 0.818 ms
4	Execution Time: 7.274 ms

Execution time using **gist index** on student_surname field: **7.274 ms**.

c. Find user by phone number

```
-- 3.
-- CREATE INDEX idx_student_phone ON student(phone_number);
explain ANALYZE select * from student where phone_number = '%';
```

a Output | Messages | Notifications

QUERY PLAN text
Index Scan using idx_student_phone on student (cost=0.13..8.14 rows=1 width=256) (actual time=0.373..0.374 rows=0 loop...
Index Cond: (phone_number = '%'::bpchar)
Planning Time: 0.211 ms
Execution Time: 0.386 ms

Execution time using **b-tree index** on student phone_number field: **0.386 ms**.

d. Find user with marks by user surname

```
-- 4.
-- CREATE INDEX idx_student_surname ON student(surname);
-- CREATE UNIQUE INDEX ON exam_result (student_id);
-- set enable_seqscan = off;
-- SET enable_hashjoin = off;
-- SET enable_mergejoin = off;
-- SET enable_nestloop = off;
explain ANALYZE select * from student join exam_result on student.id = exam_result.student_
```

ta Output | Messages | Notifications

QUERY PLAN text
Nested Loop (cost=0.25..16.29 rows=1 width=272) (actual time=0.030..0.031 rows=1 loops=1)
Join Filter: (student.id = exam_result.student_id)
-> Index Scan using idx_student_id on student (cost=0.13..8.14 rows=1 width=256) (actual time=0.022..0.023 rows=1 loops=1)
Filter: ((surname)::text = 'Kuffer'::text)
-> Index Scan using exam_result_student_id_idx on exam_result (cost=0.13..8.14 rows=1 width=16) (actual time=0.005..0.005 rows=1 loops=1)
Planning Time: 9.844 ms
Execution Time: 0.046 ms

Execution time using **b-tree index** on student surname field: **0.046 ms**.

INSERTING TEST DATA

Inserting 100000 random records into student table

```
-- generate random data

insert into student (name, surname, date_of_birth, phone_number, primary_skill)
select left(md5 (random ()::text),10), left(md5 (random ()::text),10), '2018-01-31'::date +interval '1 mons', left(md5 (random ()::text),10), left(md5 (random ()::text),10)
from generate_series (1,94999)
order by random ();
```

Data Output | Messages | Notifications

	id [PK] integer	name character varying (30)	surname character varying (30)	date_of_birth date	phone_number character (10)	primary_skill text	created_datetime timestamp without time zone	updated_datetime timestamp without time zone
1	8	59200d2da1	59200d2da1	2018-02-28	59200d2da1	59200d2da1	2023-04-20 17:30:06.562531	2023-04-20 17:30:06.562531
2	9	29eeba56b7	29eeba56b7	2018-02-28	29eeba56b7	29eeba56b7	2023-04-20 17:30:06.562531	2023-04-20 17:30:06.562531
3	10	51f60357e7	51f60357e7	2018-02-28	51f60357e7	51f60357e7	2023-04-20 17:30:06.562531	2023-04-20 17:30:06.562531
4	11	8d2f187342	8d2f187342	2018-02-28	8d2f187342	8d2f187342	2023-04-20 17:30:06.562531	2023-04-20 17:30:06.562531
5	12	b9d8b98930	b9d8b98930	2018-02-28	b9d8b98930	b9d8b98930	2023-04-20 17:30:06.562531	2023-04-20 17:30:06.562531
6	13	dd02d52e46	dd02d52e46	2018-02-28	dd02d52e46	dd02d52e46	2023-04-20 17:30:06.562531	2023-04-20 17:30:06.562531
7	14	fcfb58cd10	fcfb58cd10	2018-02-28	fcfb58cd10	fcfb58cd10	2023-04-20 17:30:06.562531	2023-04-20 17:30:06.562531
8	15	55208fb69c	55208fb69c	2018-02-28	55208fb69c	55208fb69c	2023-04-20 17:30:06.562531	2023-04-20 17:30:06.562531
9	16	7068059cd3	7068059cd3	2018-02-28	7068059cd3	7068059cd3	2023-04-20 17:30:06.562531	2023-04-20 17:30:06.562531
10	17	7f8d6dae40	7f8d6dae40	2018-02-28	7f8d6dae40	7f8d6dae40	2023-04-20 17:30:06.562531	2023-04-20 17:30:06.562531
11	18	dd8a27b12f	dd8a27b12f	2018-02-28	dd8a27b12f	dd8a27b12f	2023-04-20 17:30:06.562531	2023-04-20 17:30:06.562531
12	19	28351b2fb8	28351b2fb8	2018-02-28	28351b2fb8	28351b2fb8	2023-04-20 17:30:06.562531	2023-04-20 17:30:06.562531
13	20	fb951974e7	fb951974e7	2018-02-28	fb951974e7	fb951974e7	2023-04-20 17:30:06.562531	2023-04-20 17:30:06.562531
14	21	75bbda2cf6	75bbda2cf6	2018-02-28	75bbda2cf6	75bbda2cf6	2023-04-20 17:30:06.562531	2023-04-20 17:30:06.562531
15	22	09786d9ea2	09786d9ea2	2018-02-28	09786d9ea2	09786d9ea2	2023-04-20 17:30:06.562531	2023-04-20 17:30:06.562531
16	23	c01bde73e3	c01bde73e3	2018-02-28	c01bde73e3	c01bde73e3	2023-04-20 17:30:06.562531	2023-04-20 17:30:06.562531
17	24	a8512dc3a5	a8512dc3a5	2018-02-28	a8512dc3a5	a8512dc3a5	2023-04-20 17:30:06.562531	2023-04-20 17:30:06.562531
18	25	d3fbe314ef	d3fbe314ef	2018-02-28	d3fbe314ef	d3fbe314ef	2023-04-20 17:30:06.562531	2023-04-20 17:30:06.562531
19	26	bfd9523ca1	bfd9523ca1	2018-02-28	bfd9523ca1	bfd9523ca1	2023-04-20 17:30:06.562531	2023-04-20 17:30:06.562531

Total rows: 1000 of 100000 | Query complete 00:00:00.185 | Ln 4, Col 23

Inserting 1000 random records into subject table

```
insert into subject (subject_name, student_id)
select left(md5 (random ()::text),10), floor(random() * 100000 + 1)::int
from generate_series (1,1000)
order by random ();
```

Data Output | Messages | Notifications

	id [PK] integer	subject_name character varying (50)	student_id integer
1	2	7a35d34887	63
2	3	5e78077036	66
3	4	fb991a4679	86
4	5	38c7b01874	142
5	6	7a7119779b	298
6	7	aacee321c4	336
7	8	642c63c5d0	404
8	9	27c243b8f1	539
9	10	f27fde0a5d	544
10	11	069ace4a41	819
11	12	de7cb8b265	992
12	13	2d9186960f	1101
13	14	d07193c99e	1247
14	15	b694cfc801	1315
15	16	839190093f	1436
16	17	4d2f9fc261	1492
17	18	41e84617e8	1640
18	19	0c521544b0	2076
19	20	05a244d18c	2184

Total rows: 1000 of 1000 Query complete 00:00:00.107

Inserting 1000000 random records into exam_result table

```
insert into exam_result (student_id, subject_id, mark)
select floor(random() * 99993 + 7)::int, floor(random() * 998 + 2)::int, floor(random() * 100 + 1)::int
from generate_series (1,1000000)
order by random ();
```

Data Output | Messages | Notifications

	id [PK] integer	student_id integer	subject_id integer	mark integer
1	1	7	2	1
2	2	7	2	1
3	3	7	2	1
4	4	7	2	1
5	5	7	2	1
6	6	7	2	1
7	7	7	2	1
8	8	8	2	1
9	9	8	2	1
10	10	8	2	1
11	11	8	2	1
12	12	8	2	1
13	13	9	2	1
14	14	9	2	1
15	15	9	2	1
16	16	9	2	1
17	17	9	2	1
18	18	9	2	1
19	19	9	2	1

Total rows: 1000 of 1000000 Query complete 00:00:00.522

Results after inserting test data:

a. Find user by name

```
1  -- 1.
2  -- CREATE INDEX idx_student_name ON student(name);
3  -- CREATE INDEX idx_hash_student_name ON student USING hash(name);
4  -- set enable_seqscan = off;
5  explain ANALYZE select * from student where name = '7b28d892f4';
```

Data Output | Messages | Notifications

	QUERY PLAN text
1	Index Scan using idx_student_name on student (cost=0.42..8.44 rows=1 width=68) (actual time=0.029..0.030 rows=1 loops=1)
2	Index Cond: ((name)::text = '7b28d892f4'::text)
3	Planning Time: 0.788 ms
4	Execution Time: 0.040 ms

Execution time using **b-tree index** on student name field: **0.040 ms**.

```
1  -- 1.
2  -- CREATE INDEX idx_student_name ON student(name);
3  -- CREATE INDEX idx_hash_student_name ON student USING hash(name);
4  -- set enable_seqscan = off;
5  explain ANALYZE select * from student where name = '7b28d892f4';
```

Data Output | Messages | Notifications

	QUERY PLAN text
1	Index Scan using idx_hash_student_name on student (cost=0.00..8.02 rows=1 width=68) (actual time=0.021..0.025 rows=1 loops=1)
2	Index Cond: ((name)::text = '7b28d892f4'::text)
3	Planning Time: 0.376 ms
4	Execution Time: 0.047 ms

Execution time using **HASH index** on student name field: **0.047 ms**.

b. Find user by surname

```
7  -- 2.
8  -- CREATE EXTENSION pg_trgm;
9  -- CREATE INDEX idx_student_surname_gist ON student USING gist (surname gist_trgm_ops);
10 -- SET enable_bitmapscan TO off;
11 explain ANALYZE select * from student where surname like '7b%';
12
```

Data Output | Messages | Notifications

	QUERY PLAN text
1	Bitmap Heap Scan on student (cost=4.36..41.78 rows=10 width=68) (actual time=16.957..18.458 rows=389 loops=1)
2	Recheck Cond: ((surname)::text ~~ '7b%'::text)
3	Heap Blocks: exact=337
4	-> Bitmap Index Scan on idx_student_surname_gist (cost=0.00..4.36 rows=10 width=0) (actual time=16.912..16.912 rows=389 loop...
5	Index Cond: ((surname)::text ~~ '7b%'::text)
6	Planning Time: 8.365 ms
7	Execution Time: 19.135 ms

Execution time using **gist index** on student surname field: **19.135 ms**.

c. Find user by phone number

```
-- 3.
-- CREATE INDEX idx_student_phone ON student(phone_number);
explain ANALYZE select * from student where phone_number = '%';
```

ata Output | Messages | Notifications

QUERY PLAN text
Index Scan using idx_student_phone on student (cost=0.42..8.44 rows=1 width=68) (actual time=0.042..0.043 rows=0 loop...
Index Cond: (phone_number = '%'::bpchar)
Planning Time: 0.289 ms
Execution Time: 0.059 ms

Execution time using **b-tree index** on student phone_number field: **0.059 ms**.

d. Find user with marks by user surname

```
-- 4.
-- CREATE INDEX idx_student_surname ON student(surname);
-- CREATE INDEX idx_student_id_exam_result ON exam_result (student_id);
-- set enable_seqscan = off;
-- SET enable_hashjoin = off;
-- SET enable_mergejoin = off;
-- SET enable_nestloop = off;
explain ANALYZE select * from student join exam_result on student.id = exam_result.student_


-- generate random data
```

ta Output | Messages | Notifications

QUERY PLAN text
Nested Loop (cost=10000000000.70..10000000017.02 rows=10 width=84) (actual time=0.455..0.849 rows=12 loops=1)
-> Index Scan using idx_student_surname_gist on student (cost=0.28..8.30 rows=1 width=68) (actual time=0.390..0.781 rows=1 loops=1)
Index Cond: ((surname)::text = '23c51a5bb0'::text)
-> Index Scan using idx_student_id_exam_result on exam_result (cost=0.42..8.62 rows=11 width=16) (actual time=0.062..0.064 rows=12 loop...
Index Cond: (student_id = student.id)
Planning Time: 1.512 ms
Execution Time: 0.868 ms

Execution time using **gist index** on student surname field: **0.868 ms**.

	Before test data		After test data	
Find user by name	B-Tree index	Hash index	B-Tree index	Hash index
	1.362 ms	0.032 ms	0.040 ms	0.047 ms
Find user by surname	GIST index			
	7.274 ms		19.135 ms	
Find user by phone	B-Tree index			
	0.386 ms		0.059 ms	
Find user with marks by user surname	B-Tree index		GIST index	
	0.046 ms		0.868 ms	

Task 5. Add trigger that will update column updated_datetime to current date in case of updating any of student.

```
-- 5. Updating date_time column
-- UPDATE student SET name = 'edeaaa9c12' WHERE id = 7; --not working
-- drop  function update_student_updated_datetime() cascade;

CREATE FUNCTION update_student_updated_datetime() RETURNS TRIGGER AS $$
BEGIN
    UPDATE student SET updated_datetime=now() WHERE id=OLD.id;
    RETURN NEW;
END; $$
LANGUAGE plpgsql;

CREATE TRIGGER student_update
    after UPDATE of name ON student
    FOR EACH ROW
    EXECUTE FUNCTION update_student_updated_datetime();

UPDATE student SET name = 'edeaaa9c22' WHERE id = 8; --working
```

```
4  select * from student where id=8;
```

Data Output | Messages | Notifications

	id [PK] integer	name character varying (30)	surname character varying (30)	date_of_birth date	phone_number character (10)	primary_skill text	created_datetime timestamp without time zone	updated_datetime timestamp without time zone
1	8	edeaaa9c22	59200d2da1	2018-02-28	59200d2da1	59200d2da1	2023-04-20 17:30:06.562531	2023-04-25 23:28:54.90834

Task 6. Add validation on DB level that will check username on special characters (reject student name with next characters '@', '#', '$').

```
63  -- 6.
64  -- alter table student
65  -- add constraint name_check
66  -- CHECK (position('@' in name) = 0 AND position('#' in name) = 0 AND position('$' in name) = 0);
67
68  insert into student (name, surname, date_of_birth, phone_number, primary_skill) values ('To@m', 'Kuffer', '2003-06-30', 0993
69
```

Data Output | Messages | Notifications

```
ERROR: ПОМИЛКА:  новий рядок для відношення "student" порушує перевірне обмеження перевірку "name_check"
DETAIL:  Помилковий рядок містить (100006, To@m, Kuffer, 2003-06-30, 993784466 , signing, 2023-04-24 16:29:41.363156, 2023-04-24 16:29:41.363156).


SQL state: 23514
```

Task 7. Create snapshot that will contain next data: student name, student surname, subject name, mark (snapshot means that in case of changing some data in source table – your snapshot should not change).

```
70  -- 7. join 3 tables - snapshot
71  select student.name, student.surname, subject.subject_name, exam_result.mark from student
72  join subject on subject.student_id = student.id
73  join exam_result on exam_result.student_id = student.id;
74
```

Data Output | Messages | Notifications

	name character varying (30)	surname character varying (30)	subject_name character varying (50)	mark integer
10182	fddcb94cd3	fddcb94cd3	acf8eae036	99
10183	fddcb94cd3	fddcb94cd3	acf8eae036	99
10184	fddcb94cd3	fddcb94cd3	acf8eae036	99
10185	fddcb94cd3	fddcb94cd3	acf8eae036	99
10186	fddcb94cd3	fddcb94cd3	acf8eae036	99
10187	fddcb94cd3	fddcb94cd3	acf8eae036	99
10188	fddcb94cd3	fddcb94cd3	acf8eae036	99
10189	fddcb94cd3	fddcb94cd3	acf8eae036	99
10190	fddcb94cd3	fddcb94cd3	acf8eae036	99
10191	519e3bb496	519e3bb496	f6169ba248	100
10192	519e3bb496	519e3bb496	f6169ba248	100
10193	519e3bb496	519e3bb496	f6169ba248	100
10194	519e3bb496	519e3bb496	f6169ba248	100
10195	519e3bb496	519e3bb496	f6169ba248	100
10196	519e3bb496	519e3bb496	f6169ba248	100
10197	519e3bb496	519e3bb496	f6169ba248	100
10198	a8061c7014	a8061c7014	5a924b1197	100

Total rows: 10245 of 10245 Query complete 00:00:05.914

Task 8. Create function that will return average mark for input user.

```
76  -- 8.
77  -- drop function avg_mark;
78  CREATE FUNCTION avg_mark(id_value int) RETURNS NUMERIC AS $$
79  declare
80     average numeric;
81  BEGIN
82      SELECT AVG(exam_result.mark) into average from exam_result WHERE exam_result.student_id=id_value;
83      RETURN average;
84  END; $$
85  LANGUAGE plpgsql;
86
87  select avg_mark(75266);
88
```

Data Output | Messages | Notifications

	avg_mark numeric
1	76.0000000000000000

Task 9. Create function that will return average mark for input subject name.

```
-- 9.

CREATE FUNCTION avg_mark_subject(name_value text) RETURNS NUMERIC AS $$
declare
   average numeric;
BEGIN
    SELECT AVG(exam_result.mark) into average from exam_result
    join subject on subject.id = exam_result.subject_id
    WHERE subject.subject_name=name_value;
    RETURN average;
END; $$
LANGUAGE plpgsql;

select avg_mark_subject('7a35d34887');
```

Data Output | Messages | Notifications

	avg_mark_subject numeric
1	1.000000000000000000

Task 10. Create function that will return student at "red zone" (red zone means at least 2 marks <=3).

```
-- 10.
-- drop function red_zone;

CREATE FUNCTION red_zone() RETURNS table (
        id int,
        name character varying(30),
        surname character varying(30),
        date_of_birth date,
        phone_number character(10),
        primary_skill text,
        created_datetime timestamp without time zone,
        updated_datetime timestamp without time zone
    ) AS $$
BEGIN

return query
select * from student where student.id in (select student_id a from exam_result where mark <= 3 group by student_id having count(mark)>=2);

END; $$
LANGUAGE plpgsql;

select * from red_zone();
```

Data Output | Messages | Notifications

	id integer	name character varying	surname character varying	date_of_birth date	phone_number character	primary_skill text	created_datetime timestamp without time zone	updated_datetime timestamp without time zone
1	9	29eeba56b7	29eeba56b7	2018-02-28	29eeba56b7	29eeba56b7	2023-04-20 17:30:06.562531	2023-04-20 17:30:06.562531
2	10	51f60357e7	51f60357e7	2018-02-28	51f60357e7	51f60357e7	2023-04-20 17:30:06.562531	2023-04-20 17:30:06.562531
3	11	8d2f187342	8d2f187342	2018-02-28	8d2f187342	8d2f187342	2023-04-20 17:30:06.562531	2023-04-20 17:30:06.562531
4	12	b9d8b98930	b9d8b98930	2018-02-28	b9d8b98930	b9d8b98930	2023-04-20 17:30:06.562531	2023-04-20 17:30:06.562531
5	13	dd02d52e46	dd02d52e46	2018-02-28	dd02d52e46	dd02d52e46	2023-04-20 17:30:06.562531	2023-04-20 17:30:06.562531

Total rows: 2996 of 2996 | Query complete 00:00:00.227 | Ln 132, Col 23