Databases Final Project

Eric D. Stevens

Git Repo: https://github.com/Eric-D-Stevens/Employee_DB_Extension_PostgreSQL

Outline

Background: General information regarding the project
The Story: A playful story that allowed me to conceptualize the project
Data Definition: The extension to the database schema
Data Engineering: Python generation of dummy data to allow for testing
20 Questions: 10 queries on the large original database, 10 leveraging the extension

Background

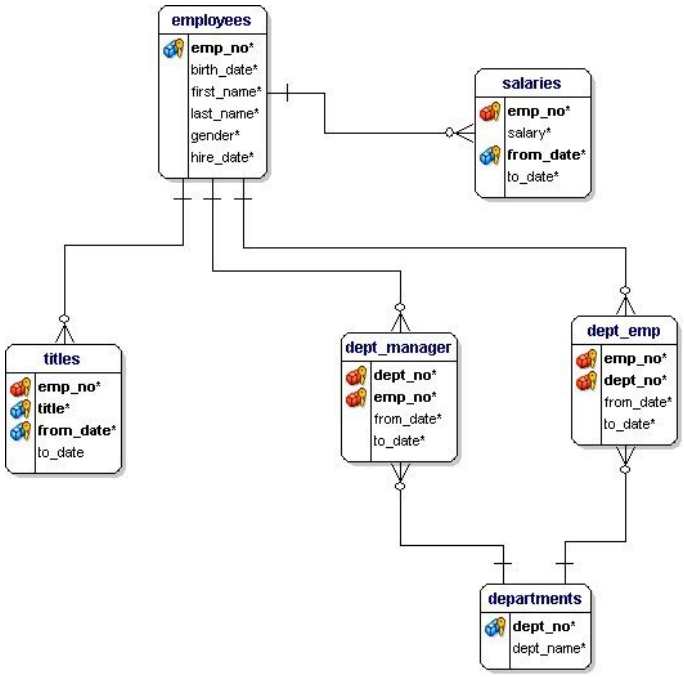
This project starts out with a very large pre-built database that I imported to PSU server. I felt that integrating with a pre-existing database would be a very good exercise. I will not be including the full database in my git repo because it is several hundred megabytes and easily accessible elsewhere: The PostgreSQL implementation can be found here: https://github.com/vrajmohan/pgsql-sample-data/tree/master/employee. This repository is a port to Postgres from the original MySql implementation which can be found here: https://github.com/datacharmer/test_db.

This project is focused on extending the existing schema with rational components that could hypothetically extend the functionality of the database. Much of the time spent on this project was focused on the effort to synthesize data that would result in logical relationships in the schema. The amount of time and effort that went into engineering the synthetic data may have taken away from implementing a more sophisticated database, the overall product of the work performs well and accomplishes what it intends to.

The story

I am working for a large software company that has a huge database of its employees, which departments they work for, what their salaries are and who the managers of each department are. The company, however, is not keeping track of the state of it's projects and the output of individual employees. As a result, the company believes it is operating inefficiently and is losing money.

The executives have contracted me to build out their database to provide more insight into the performance at the project management level, as well as at individual contributor level. To do this they have agreed to build out a front end system that will allow employees to summarize what they have been working on in a given time period. They want me to build out a back end to store the employee self reporting records, as well as a number of other organizational relations that will help them locate inefficiencies in their company. The database schema that they currently have can be seen below.



Data Definition

The DDL file for the database extension can be found at the link below.

https://github.com/Eric-D-Stevens/Employee_DB_Extension_PostgreSQL/blob/master/ddl.sql

In order to organize the information in a way that the company executives will be able to gether insight from, a schema will be created alongside the current one and attach to it. There will be five tables added to the database:

1. Project: The project table will hold the project name, the department responsible for it, and the project manager. The department will be referenced to the department table in the original database and the project manager will reference the employees table in the original database. (NOTE: the assumption is that project managers and department managers are not the same.)

```
-- PROJECTS TABLE --
CREATE TABLE employee.projects
(
    proj_id INTEGER NOT NULL,
    proj_name varchar(50) NOT NULL,
    dept_id CHARACTER(4) NOT NULL,
    prj_mngr INTEGER NOT NULL,
    CONSTRAINT projects_pkey PRIMARY KEY (proj_id),
    CONSTRAINT projects_dept_id_fkey FOREIGN KEY (dept_id)
        REFERENCES employee.departments (dept_no),
    CONSTRAINT projects_prj_mngr_fkey FOREIGN KEY (prj_mngr)
        REFERENCES employee.employees (emp_no)
);
```

2. Tasks: Projects are made up of tasks, so the project table has a one to many relationship with the tasks table. Each task has a task ID and also maintains the ID of the project it belongs to. Each task will also have a task manager, who may or may not be the project manager.

```
-- TASKS TABLE --
CREATE TABLE employee.tasks
(
    task_id INTEGER NOT NULL,
    project_id INTEGER NOT NULL,
    task_mngr INTEGER NOT NULL,
    CONSTRAINT tasks_pkey PRIMARY KEY (task_id),
    CONSTRAINT tasks_project_id_fkey FOREIGN KEY (project_id)
        REFERENCES employee.employees (emp_no)
);
```

3. Task_Team: This table will keep track of which employees are assigned to which task. Individual employees can be assigned to multiple tasks and there can be many employees assigned to each task, making this table a many to many relationship representation. The table simply has a task ID and an employee id as a compound

primary key. Each of the attributes is a foregin key to its respective table.

```
35  -- TASK_TEAM TABLE --
36  CREATE TABLE employee.task_team
37  (
38      task_id INTEGER NOT NULL,
39      emp_no INTEGER NOT NULL,
40      CONSTRAINT task_team_pkey PRIMARY KEY (task_id, emp_no),
41      CONSTRAINT task_team_task_id_fkey FOREIGN KEY (task_id)
42          REFERENCES employee.tasks (task_id),
43      CONSTRAINT task_team_emp_no_fkey FOREIGN KEY (emp_no)
44          REFERENCES employee.employees (emp_no)
45  );
```

4. Work_Log: The work log table is the one that the company is developing a front end for. It is where employees will summarize what they have been doing over a given period of time. When They enter a summary of what they have been working on, they will be entering it with respect to a task. Therefore, the work log table has both an employee ID, and a task ID as foreign keys. Each log has its own ID, space for a 1000 character summary of the work performed, a start timestamp and an end timestamp.

```
47  -- WORK_LOG TABLE --
48  CREATE TABLE employee.work_log
49  (
50      log_id INTEGER NOT NULL,
51      emp_no INTEGER NOT NULL,
52      task_id INTEGER NOT NULL,
53      summary varchar(1000),
54      start_timestamp TIMESTAMP NOT NULL,
55      end_timestamp TIMESTAMP NOT NULL,
56      CONSTRAINT work_log_pkey PRIMARY KEY (log_id),
57      CONSTRAINT work_log_emp_no_fkey FOREIGN KEY (emp_no)
58          REFERENCES employee.employees (emp_no),
59      CONSTRAINT work_log_task_id_fkey FOREIGN KEY (task_id)
60          REFERENCES employee.tasks (task_id)
61  );
```

Data Engineering

Once the schema is set up we can assume it is ready to go. Projects can be entered by the department managers and tasks and work logs can be filled out further down the corporate ladder. However, it would be very nice to test our system before we put it in production. This is where things get complicated.

Since work logs and project records have not been kept track of so far, we need to synthesize artificial data. This is not a simple task, since there is a logical organization to the way projects and tasks are partitioned amongst departments. Data cannot just be entered randomly to get a sense of whether the system works. The data will be synthesized sequentially using Python and the Python PSQL library PsycoPG2.

PsycoPG2 Helper Class

A helper class was implemented to maintain database credentials, allow for single call SQL queries, and create utilities for common SQL queries that were used to generate the random data, such as getting a single column from a table. The link to this helper class can be found with the link below.

https://github.com/Eric-D-Stevens/Employee_DB_Extension_PostgreSQL/blob/master/data_engineering/PG.py

Creating the Project Table

The project table was created by utilizing the names of the missions in the Spy database. Each project is given a name from the spy.mission table and assigned to a random department. Only specific “senior” job titles are allowed to be project managers, so the employee.title table is parsed through to find employee IDs that have the senior titles. They are randomly assigned as the project managers in the projects table. The code that created the data for this table can be found with the link below.

https://github.com/Eric-D-Stevens/Employee_DB_Extension_PostgreSQL/blob/master/data_engineering/create_project.py

Creating the Tasks Table

The tasks table holds tasks that are part of a project. A Project can have many tasks and a task may belong to only one project. Tasks also have managers that must come from the same

department as the project is assigned to. The code for generating the task table can be found with the link below.

https://github.com/Eric-D-Stevens/Employee_DB_Extension_PostgreSQL/blob/master/data_engineering/create_tasks.py

Creating the Task Team Table

The task team table is the relational table between collections of employees and tasks. Employees can be on multiple task teams, as long as all those tasks belong to projects under their assigned department. The code that creates the task team table can be found with the link below.

https://github.com/Eric-D-Stevens/Employee_DB_Extension_PostgreSQL/blob/master/data_engineering/create_task_team.py

Creating the Work Log table

The work log table represents employees reporting on what they have been doing. Each work log contains an employee id, a task id, a start and end time stamp, and a summary that is generated from random characters. The code used to create the work log table can be found with the link below.

https://github.com/Eric-D-Stevens/Employee_DB_Extension_PostgreSQL/blob/master/data_engineering/create_worklog.py

20 Questions

The first 10 questions are general queries of the original database. The second 10 queries leverage the extension described in this assignment. The SQL file containing these queries can be found with the link below.

https://github.com/Eric-D-Stevens/Employee_DB_Extension_PostgreSQL/blob/master/questions.sql

1. What is the gender breakdown of the company?

```
SELECT
      ROUND( 100 *
            (SELECT COUNT(*)::NUMERIC
                  FROM employee.employees e
                        WHERE e.gender = 'F')/
            (SELECT COUNT(*)::NUMERIC
                  FROM employee.employees)
            ,2 ) as Percent_Female,
      ROUND( 100 *
            (SELECT COUNT(*)::NUMERIC
                  FROM employee.employees e
                        WHERE e.gender = 'M')/
            (SELECT COUNT(*)::NUMERIC FROM employee.employees)
            ,2 ) as Percent_Male
```

	percent_female numeric	percent_male numeric
1	40.01	59.99

2. Who are the highest and lowest paid employees?

```
SELECT e.emp_no, e.first_name, e.last_name, sm.salary
FROM employee.employees e
JOIN (
            SELECT s.emp_no, s.salary
            FROM employee.salaries s
            WHERE s.salary = (SELECT MIN(s.salary)
                  FROM employee.salaries s)
            OR s.salary = (SELECT MAX(s.salary)
                  FROM employee.salaries s)
       ) sm
ON e.emp_no = sm.emp_no
```

	emp_no integer	first_name character varying (14)	last_name character varying (16)	salary integer
1	43624	Tokuyasu	Pesch	158220
2	253406	Olivera	Baek	38623

3. How many people currently work each job title in the company?

```
SELECT t.title, COUNT(*) AS employee_count
      FROM employee.titles t
      WHERE t.to_date > current_date
      GROUP BY t.title
```

	title character varying (50)	employee_count bigint
1	Assistant Engineer	3588
2	Engineer	30983
3	Manager	9
4	Senior Engineer	85939
5	Senior Staff	82024
6	Staff	25526
7	Technique Leader	12055

4. What is the average salary for each job title?

```
SELECT t.title, ROUND(AVG(s.salary),2) as average_salary
      FROM employee.titles t
      JOIN employee.salaries s
      ON t.emp_no = s.emp_no
      GROUP BY t.title
      ORDER BY average_salary
```

	title character varying (50)	average_salary numeric
1	Technique Leader	59294.37
2	Assistant Engineer	59304.99
3	Engineer	59508.04
4	Senior Engineer	60543.22
5	Manager	66924.27
6	Staff	69309.10
7	Senior Staff	70470.84

5. How much do department managers make in salary relative to their employees?

```
SELECT d.dept_name,
      ROUND(s.salary,2) AS manager_salary,
      ROUND(avg_emp_sal.average_employee_salary,2)
            AS average_employee_salary
FROM employee.departments d
JOIN employee.dept_manager dm
ON d.dept_no = dm.dept_no
JOIN employee.salaries s
ON s.emp_no = dm.emp_no
JOIN
      (
            SELECT adepts.dept_name, AVG(asal.salary)
                  as average_employee_salary
            FROM employee.salaries asal
            JOIN employee.dept_emp aemp
            ON asal.emp_no = aemp.emp_no
            JOIN employee.departments adepts
            ON aemp.dept_no = adepts.dept_no
            GROUP BY adepts.dept_name
      ) avg_emp_sal
ON d.dept_name = avg_emp_sal.dept_name
WHERE dm.to_date > current_date
AND s.to_date > current_date;
```

	dept_name character varying (40)	manager_salary numeric	average_employee_salary numeric
1	Marketing	106491.00	71913.20
2	Finance	83457.00	70489.36
3	Human Resources	65400.00	55574.88
4	Production	56654.00	59605.48
5	Development	74510.00	59478.90
6	Quality Management	72876.00	57251.27
7	Sales	101987.00	80667.61
8	Research	79393.00	59665.18
9	Customer Service	58745.00	58770.37

6. How long did the current manager of the each department work for the company before becoming a manager?

```
SELECT current_managers.dept_name AS department,
joined_company.start_day AS started_at_company,
current_managers.from_date AS started_as_manager,
make_interval(years =>
      ((current_managers.from_date
            - joined_company.start_day)/365))
AS years_working_before_manager
FROM
      (
            SELECT  dm.dept_no, depts.dept_name,
                  dm.emp_no, dm.from_date
            FROM employee.dept_manager dm
            JOIN employee.departments depts
            ON depts.dept_no = dm.dept_no
            WHERE dm.to_date > current_date
      ) current_managers
JOIN
      (
            SELECT t.emp_no, MIN(t.from_date) as start_day
            FROM employee.titles t
            JOIN employee.dept_manager dm
            ON t.emp_no = dm.emp_no
            GROUP BY t.emp_no
      ) joined_company
ON current_managers.emp_no = joined_company.emp_no;
```

	department character varying (40)	started_at_company date	started_as_manager date	years_working_before_manager interval
1	Marketing	1986-04-12	1991-10-01	5 years
2	Finance	1985-01-14	1989-12-17	4 years
3	Human Resources	1985-08-04	1992-03-21	6 years
4	Production	1992-02-05	1996-08-30	4 years
5	Development	1986-10-21	1992-04-25	5 years
6	Quality Management	1989-06-09	1994-06-28	5 years
7	Sales	1986-12-30	1991-03-07	4 years
8	Research	1988-01-31	1991-04-08	3 years
9	Customer Service	1989-07-10	1996-01-03	6 years

7. How many employees have held 3 different titles at the company?

```
SELECT title_count.num_titles, COUNT(title_count.emp_no)
FROM
      (
            SELECT t.emp_no, COUNT(t.title) as num_titles
            FROM employee.titles t
            GROUP BY t.emp_no
            ORDER BY num_titles DESC
      ) title_count
WHERE title_count.num_titles = 3
GROUP BY title_count.num_titles;
```

	num_titles bigint	count bigint
1	3	3014

8. How many employees are named Eric?

```
SELECT COUNT(*) as erics_in_company
FROM employee.employees e
WHERE e.first_name = 'Eric';
```

	erics_in_company bigint
1	244

9. How many people currently work in each department?

```
SELECT d.dept_name, COUNT(de.emp_no) as employee_count
FROM employee.departments d
JOIN employee.dept_emp de
ON d.dept_no = de.dept_no
WHERE de.to_date > current_date
GROUP BY d.dept_name;
```

	dept_name character varying (40)	employee_count bigint
1	Customer Service	17569
2	Development	61386
3	Finance	12437
4	Human Resources	12898
5	Marketing	14842
6	Production	53304
7	Quality Management	14546
8	Research	15441
9	Sales	37701

10. How much does the company currently pay for all salaries?

```
SELECT SUM(s.salary) AS cost_of_salaries
FROM employee.salaries s
WHERE s.to_date > current_date;
```

	cost_of_salaries bigint
1	17291866123

11. How many different projects does each department have going?

```
SELECT d.dept_name, COUNT(p.proj_id) as project_count
FROM employee.departments d
JOIN employee.projects p
ON d.dept_no = p.dept_id
GROUP BY d.dept_name;
```

	dept_name character varying (40)	project_count bigint
1	Marketing	41
2	Finance	50
3	Development	50
4	Customer Service	50
5	Production	43
6	Research	54
7	Human Resources	36
8	Quality Management	44
9	Sales	36

12. What are the names of the projects under the Human Resources department?

```
SELECT d.dept_name, p.proj_name
FROM employee.departments d
JOIN employee.projects p
ON d.dept_no = p.dept_id
WHERE d.dept_name = 'Human Resources';
```

	dept_name character varying (40)	proj_name character varying (50)
1	Human Resources	Crystal Crown
2	Human Resources	Swertings
3	Human Resources	Galpsi
4	Human Resources	Merethrond
5	Human Resources	Biggie Smalls
6	Human Resources	Derndingle
7	Human Resources	Minas Anor
8	Human Resources	Red Arrow
9	Human Resources	Tupac
10	Human Resources	Overhill
11	Human Resources	Underharrow
12	Human Resources	Dimrill Stair
13	Human Resources	Deeping Wall
14	Human Resources	Fair Folk
15	Human Resources	Court of the Fountai
16	Human Resources	Desk Top

(36 Rows)

13. Which employees are both department managers and project managers?

```
SELECT dm.emp_no, dm.dept_no, p.proj_id
FROM employee.dept_manager dm
JOIN employee.projects p
ON dm.emp_no = p.prj_mngr;
```

emp_no integer	dept_no character (4)	proj_id integer

(0 rows)

14. Which employees are both project managers and task managers?

```
SELECT e.first_name, e.last_name, p.proj_name, t.task_id
FROM employee.tasks t
JOIN employee.projects p
ON t.task_mngr = p.prj_mngr
JOIN employee.employees e
ON t.task_mngr = e.emp_no
WHERE p.proj_id = t.project_id
```

	first_name character varying (14)	last_name character varying (16)	proj_name character varying (50)	task_id integer
1	Arif	Merlo	Banana	6808

15. What are the three most common job titles for project managers?

```
SELECT t.title, COUNT(*) AS title_count
FROM employee.titles t
JOIN employee.projects p
ON t.emp_no = p.prj_mngr
GROUP BY t.title
LIMIT 3;
```

	title character varying (50)	title_count bigint
1	Senior Staff	267
2	Technique Leader	87
3	Senior Engineer	50

```
-- Creation of index for fast lookup of work performed in given range
CREATE INDEX work_log_timestamp_range_idx ON
employee.work_log(start_timestamp, end_timestamp)
```

16. How many work logs have been submitted so far this year?

```
SELECT COUNT(*) AS submitted_work_logs_2020
FROM employee.work_log wl
WHERE wl.start_timestamp > '2020-01-10'::DATE;
```

	submitted_work_logs_2020 bigint
1	60530

17. Who are the 10 star employees of 2019 in terms of hours logged (most hours logged in 2019)?

```
SELECT e.emp_no, e.first_name, e.last_name, tlog.time_logged
FROM
      (
            SELECT wl.emp_no, SUM(wl.end_timestamp-wl.start_timestamp)
                  as time_logged
            FROM employee.work_log wl
            WHERE wl.start_timestamp >= '2019-01-01'::DATE
            AND wl.end_timestamp < '2020-01-10'::DATE
            GROUP BY wl.emp_no
      ) tlog
JOIN employee.employees e
ON tlog.emp_no = e.emp_no
ORDER BY tlog.time_logged DESC
LIMIT 10;
```

	emp_no [PK] integer	first_name character varying (14)	last_name character varying (16)	time_logged interval
1	279196	Valery	Fandrianto	676 days 19:00:00
2	437163	Boguslaw	Busillo	593 days 22:00:00
3	16811	Baziley	Krichel	587 days 12:00:00
4	469064	Huai	Linares	578 days 17:00:00
5	429745	Krister	Boguraev	577 days 22:00:00
6	438468	Peternela	Socorro	576 days 17:00:00
7	414597	Tonia	Ressouche	565 days 13:00:00
8	75160	Marit	Mandelberg	562 days 24:00:00
9	204446	Chiradeep	Kroft	561 days 27:00:00
10	468837	Lillian	Slobodova	548 days 22:00:00

Note: This is obviously where the randomly generated data had some shortcomings, it is not possible for someone to log 2 years worth of hours.This is because in the generation of data employees were not restricted from working on multiple tasks in parallel. Also, the work log allowed entries to span days and even months, so this question does not make 100 percent sense for the design of the data.

18. How much time has been put into each task in the 'Dark Years' project?

```
SELECT dark_year.proj_name, dark_year.task_id,
sum(wl.end_timestamp - wl.start_timestamp) time_on_task
FROM
      (
            SELECT p.proj_id, p.proj_name, t.project_id, t.task_id
            FROM employee.projects p
            JOIN employee.tasks t
            ON p.proj_id = t.project_id
            WHERE p.proj_name = 'Dark Years'
      ) dark_year
JOIN employee.work_log wl
ON dark_year.task_id = wl.task_id
GROUP BY (dark_year.proj_name, dark_year.task_id)
ORDER BY time_on_task DESC;
```

1	Dark Years	801	20736 days 580:0...
2	Dark Years	809	19809 days 545:0...
3	Dark Years	800	18762 days 514:0...
4	Dark Years	816	18721 days 518:0...
5	Dark Years	798	17918 days 541:0...
6	Dark Years	824	17490 days 534:0...
7	Dark Years	822	17327 days 493:0...
8	Dark Years	811	17067 days 519:0...
9	Dark Years	823	15781 days 442:0...
10	Dark Years	820	15546 days 500:0...
11	Dark Years	813	14696 days 374:0...
12	Dark Years	804	13930 days 415:0...

(33 Rows)

19. Which employees worked on the 'Dark Years' project in June of 2018?

```
SELECT e.emp_no, e.first_name, e.last_name
FROM employee.employees e
JOIN employee.work_log wl
ON wl.emp_no = e.emp_no
WHERE wl.start_timestamp >= '2018-06-01'::DATE
AND wl.end_timestamp < '2018-07-01'::DATE
AND wl.task_id IN
      (
            SELECT t.task_id
            FROM employee.tasks t
            JOIN employee.projects p
            ON p.proj_id = t.project_id
            WHERE p.proj_name = 'Dark Years'
      )
```

	emp_no [PK] integer	first_name character varying (14)	last_name character varying (16)
1	265950	Eishiro	Schicker
2	58676	Mitsuyuki	Nittel
3	47644	Rajmohan	Zlotek
4	278010	Soenke	Falster
5	20299	Chaitali	Fortenbacher

20. A system crash happened on New Years 2020 at around 4:00pm! Which employees logged work on that day in the hours adjacent to the crash?

```
SELECT e.emp_no, e.first_name, e.last_name,
      l.end_timestamp AS commited,
      l.summary AS summary_of_work
FROM employee.employees e
JOIN employee.work_log l
ON e.emp_no = l.emp_no
WHERE l.end_timestamp > '2020-01-01 15:00:00'::TIMESTAMP
AND l.end_timestamp < '2020-01-01 17:00:00'::TIMESTAMP;
```

	emp_no integer	first_name character varying (14)	last_name character varying (16)	commited timestamp without time zone	summary_of_work character varying (1000)
1	66600	Tetsurou	Huxford	2020-01-01 16:00:00	Obg wGaaSBovyoZwXFOf I p i...
2	55747	Roselyn	Ghandeharizadeh	2020-01-01 16:00:00	p MqfEpaJ wDFv ViW b If vZ F...
3	259921	Froduald	Panwar	2020-01-01 16:00:00	zkXuuHVdh gQowr BGbAy xgr...
4	297446	Sasan	Marquardt	2020-01-01 16:00:00	CyHYmPhIdfwYx to cpuoX T fX...
5	100938	Jiafu	Bennis	2020-01-01 16:00:00	uVu Ml BJhzyL rg xfnvmMSJJ...
6	276400	Utpal	Pouyioutas	2020-01-01 16:00:00	QTuXzCep k RSj uQu KLtSOgm...
7	79527	Boutros	Schwabacher	2020-01-01 16:00:00	w OJdiGXivD SgGRtSFaotfhBX...
8	84878	Sadun	Ratnakar	2020-01-01 16:00:00	ImoFMhS AosiBFYqwMgU qgW...
9	103339	Geoffrey	Worfolk	2020-01-01 16:00:00	pQcsCZfk RUFiFG Y V er uxn i...
10	65347	Shichao	Peltason	2020-01-01 16:00:00	dEiG Um zkVOq PtZjbuRQqIN c...
11	258567	Masali	Peck	2020-01-01 16:00:00	esrwJxTFfuoBgzSQ AMqontOq...
12	469861	Matt	Baer	2020-01-01 16:00:00	E LU DvYmFVLakGrpr MT FZCe...
13	270518	Gererd	Setlzner	2020-01-01 16:00:00	BgDmAAYb AYsjIGmvk Kcp B...
14	91434	Uno	Toyoshima	2020-01-01 16:00:00	V jVjPDgLrEq L XfWndoSk Dijs ...
15	99946	Mostafa	Aloisi	2020-01-01 16:00:00	mspyULqb kgLgtib NBiNqvtMs...
16	22464	Ingemar	Skrikant	2020-01-01 16:00:00	pdoc ayOKo YIxDiMkkMkZqXo...