SQL Basics Part IV - Advanced SQL Commands

26.10.2019

SQL Basics

TIMESTAMPS & EXTRACT

★SQL allows us to use the **timestamp data type** to retain time information.
★Later we will learn how to create timestamp columns. But for now we will focus on extracting information from timestamp objects.

★The Extract function allows us to extract parts from a date.
★We can extract many types of time-based information from a timestamp.

`Example:`

`★We know that paym+ent table has a column named payment_date and it stores values of timestamp type. Let's extract data from this table.`

```
SELECT customer_id, extract(day from payment_date)
FROM payment;
```

`★Now I've extracted the day info from the payment_date column, meaning I have a`

column of days now.
★Temelde extract sadece belirtilen timestamp tipindeki column'u değiştiriyor. Mesela tam tarih yerine sadece ay veya gün'e çevirebiliriz. Google'a bakarsak timestamp tipinden daha bir çok farklı info extract edebileceğimizi görürüz.

MATHEMATICAL OPERATORS & FUNCTIONS

★SQL comes with a lot of mathematical operators built-in that are very useful for numeric column types.
★You can explore all of them in this link: https://www.postgresql.org/docs/9.5/functions-math.html

Example:

★Let's say I want to obtain a new_id column using customer_id and rental_id columns:

```
SELECT customer_id + rental_id AS new_id
FROM payment;
```

★So we've added two columns, in the same way we could've multiplied, subtract, divide, etc. them.

```
SELECT AVG(amount)
FROM payment;
```

★We can also use mathematical functions for the columns.

STRING OPERATORS & FUNCTIONS

★SQL comes with a lot of string operators built-in.
★You can explore all of them in this link: https://www.postgresql.org/docs/9.5/functions-string.html

Example:

★Let's try string concatenation for two string typed columns:

```
SELECT first_name || ' ' || last_name AS full_name
FROM customer;
```

★Now we have the full_name column which includes first_name + ' ' + last_name.

```
SELECT first_name, char_length(first_name)
FROM customer;
```

★We can also use string functions for the columns. So it returns 2 columns: first_name and number of chars of the first_name.

```
SELECT first_name, lower(first_name)
FROM customer;
```

★lower() returns lowercase strings.

SUBQUERY

★A subquery allows us to use multiple SELECT statements, where we basically have a query within a query.

★Let's better understand it using an example:

- Suppose we want to find the films whose rental rate is higher than the average rental rate.

★One way to do it is to use 2 seperate select statements:

Example:

```
SELECT AVG(rental_rate)
FROM film;
```

★Now we've find the the average rental_rate as $2.98. We will use this value in the second query:

```
SELECT title, rental_rate
FROM film
WHERE rental_rate > 2.98;
```

★So the way explained above is not the best, because we need to write down the result of the first query. So let's try subquery method:

```
Example:

SELECT film_id, rental_rate
FROM film
WHERE rental_rate > (SELECT AVG(rental_rate) FROM film);
```

★The statement: `(SELECT AVG(rental_rate) FROM film)` is a subquery. First the subquery is executed and then the rest of the code will work as same.

★ Note that, the subquery may return a set with more than 1 elements as well. If it is the case we need to use the WHERE statement accordingly.

★ Also we can use more complex subqueries:

```
Example:

SELECT inventory.film_id
FROM rental
INNER JOIN inventory ON inventory.inventory_id = rental.inventory_id
WHERE return_date BETWEEN '2005-05-09' AND '2005-05-30';
```

★So we're joining the inventory table with the rental table on the inventory_id. Then we select the specific film_ids. The result will be a film_id set.

★Now we are gonna use this whole query as a subquery to get the titles of the films:

```
SELECT film_id,title
FROM film
WHERE film_id IN

(SELECT inventory.film_id
FROM rental
INNER JOIN inventory ON inventory.inventory_id = rental.inventory_id
WHERE return_date BETWEEN '2005-05-09' AND '2005-05-30');
```

★So we are treating the subquery as a list of values and check if there is a match with these values in our film table.

SELF JOIN

★We've learned how to join a table to the other tables using INNER JOIN, LEFT OUTER JOIN or RIGHT OUTER JOIN statements.
★There is a special case that we join a table to itself, which is known as self join.

★We use self join when we want to combine rows with other rows in the same table.
★To perform the self join operation, we must use a table alias to help SQL distunguish the left table from the right table of the same table.

★Let's say I have table as below:

employee_name	**employee_location**
Joe	New York
Sunil	India
Alex	Russia
Albert	Canada
Jack	New York

★Suppose we want to find out which employees are from the same location as the employee named Joe.

★One way to do so would be:

```
Example:

SELECT employee_name
FROM employee
WHERE employee_location = 'New York'
```

★Another way would be using a subquery:

```
Example:

SELECT employee_name
FROM employee
WHERE employee_location IN
(SELECT employee_location FROM employee WHERE employee_name = "Joe");
```

★While the subquery is a valid solution, it is actually more efficient to use a self join, where we join a table to itself:

```
Example:

SELECT e1.employee_name
FROM employee AS e1, employee AS e2
WHERE e1.employee_location = e2.employee_location AND e2.employee_name = 'Joe';

Alternative way to write it down would be:

SELECT e1.employee_name
FROM employee AS e1
JOIN employee AS e2
ON e1.employee_location = e2.employee_location AND e2.employee_name = 'Joe';
```

★Table e1 and e2 will be the same:

employee_name	employee_location
Joe	New York
Sunil	India
Alex	Russia
Albert	Canada
Jack	New York

★The result of the self join will be:

e1.employee_name	e1.employee_location	e2.employee_name	e2.employee_location
Joe	New York	Joe	New York
Jack	New York	Joe	New York

EXCERCISES

★Assume we have 3 tables: **cd.bookings**, **cd.facilities**, **cd.members**

bookings: bookid facid memid starttime slots
facilities: facid name membercost guestcost initialoutlay monthlymaintanence
bookings: memid surname firstname address zipcode telephone recommendedby joindate

---

Example:

★How can you produce a list of facilities that charge a fee to members, and that fee is less than 1/50th of the monthly maintenance cost? Return the facid, facility name, member cost, and monthly maintenance of the facilities in question.

```
SELECT facid, name, membercost, monthlymaintenance
FROM cd.facilities
WHERE membercost > 0 AND (membercost < monthlymaintenance/50.0);
```

Example:

★How can you retrieve the details of facilities with ID 1 and 5? Try to do it without using the OR operator.

```
SELECT *
FROM cd.facilities
WHERE facid IN (1,5)
```

Example:

★How can you produce a list of members who joined after the start of September 2012? Return the memid, surname, firstname, and joindate of the members in question.

```
SELECT memid, surname, firstname, joindate
FROM cd.members
WHERE joindate >= '2012-09-01';
```

Example:

★How can you produce an ordered list of the first 10 surnames in the members table? The list must not contain duplicates.

```
SELECT DISTINCT surname
FROM cd.members
ORDER BY surname
LIMIT 10;
```

Example:

★ You'd like to get the signup date of your last member. How can you retrieve this information?

```
SELECT *
FROM cd.members
ORDER BY joindate DESC
LIMIT 1;
```

An alternative way:

```
SELECT MAX(joindate)
FROM cd.members;
```

Example:

★Produce a count of the number of facilities that have a cost to guests of 10 or more.

```
SELECT COUNT(*)
FROM cd.facilities
WHERE guestcost>=10;
```

Example:

★Produce a list of the total number of slots booked per facility in the month of September 2012. Produce an output table consisting of facility id and slots, sorted by the number of slots.

```
SELECT facid, SUM(slots) AS totalslots
FROM cd.bookings
WHERE starttime BETWEEN '2012-09-01' AND '2012-10-01'
```

```
GROUP BY facid
ORDER BY totalslots
```

Example:

★ How can you produce a list of the start times for bookings for tennis courts, for the date '2012-09-21'? Return a list of start time and facility name pairings, ordered by the time.

```
SELECT cd.facilities.name, cd.bookings.starttime
FROM cd.bookings
INNER JOIN cd.facilities ON cd.facilities.facid = cd.bookings.facid
WHERE cd.bookings.facid IN (0,1) AND starttime BETWEEN '2012-09-21' AND '2012-09-22';
```

Example:

★ How can you produce a list of the start times for bookings by members named 'David Farrell'?

```
SELECT cd.bookings.starttime, cd.members.firstname, cd.members.surname
FROM cd.bookings
INNER JOIN cd.members ON cd.bookings.memid = cd.members.memid
WHERE cd.members.firstname = 'David' AND cd.members.surname='Farrell'
```

More blogs