Overview**:**

- Timestamps and EXTRACT
- Math Functions
- String Functions
- Sub-query
- Self-Join

Timestamps and EXTRACT (Part 1): Displaying current time information

- **TIME** – Contains only time
- **DATE** – Contains only date
- **TIMESTAMP** – Contains date and time
- **TIMESTAMPTZ** – Contains date, time, and time zone

Query:

SHOW ALL
SHOW TIMEZONE # US/Easten
SELECT NOW () # timestamp with time zone (date + time)
SELECT TIMEOFDAY () # string (Thu Mar 19 14:23:26)
SELECT CURRENT_TIME # timestamp with time zone
SELECT CURRENT_DATE # date

Timestamps and EXTRACT (Part 2): Extracting time and date information

- **EXTRACT ()**
  - **YEAR**
    - **SELECT EXTRACT (YEAR FROM** date_col) **FROM** Table
  - **MONTH**
  - **DAY**
  - **WEEK**
  - **QUARTER**
- **AGE ()**: calculates and returns the current age
  - **SELECT AGE (**date_col**) FROM** Table => 13 years 1 mon 5 days 01:34:13
- **TO_CHAR ()**: general function to convert date types to text, useful for timestamp formatting, also can convert integer to string
  - **SELECT TO_CHAR (**date_col, 'mm-dd-yyyy'**) FROM** Table
  - https://www.postgresql.org/docs/12/functions-formatting.html

Timestamps and Extract Challenge Tasks

1. During which months did payments occur? Format your answer to return back the full month name.

   My answer:
   SELECT DISTINCT EXTRACT (MONTH FROM payment_date**)**
   FROM payment

	date_part double precision
1	5
2	2
3	4
4	3

   Hints: don't need to use EXTRACT for this query

   Revised:
   SELECT DISTINCT TO_CHAR (payment_date, 'MONTH'**)**
   FROM payment

	to_char text
1	MARCH
2	MAY
3	FEBRUARY
4	APRIL

2. How many payments occurred on a Monday? NOTE: We didn't show you exactly how to do this but use the documentation or Google to figure this out!
   Hints:
   - Use EXTRACT
   - Review the **dow** keyword
   - PostgreSQL considers Sunday the start of a week (index at 0)

   SELECT COUNT (*) FROM payment
   WHERE EXTRACT (dow **FROM** payment_date**)** = 1

Mathematical Functions and Operators

- Operators
  SELECT ROUND (rental_rate/replacement_cost, 2) **FROM** film

Operator	Description	Example	Result
+	addition	2 + 3	5
-	subtraction	2 - 3	-1
*	multiplication	2 * 3	6
/	division (integer division truncates the result)	4 / 2	2
%	modulo (remainder)	5 % 4	1
^	exponentiation (associates left to right)	2.0 ^ 3.0	8
|/	square root	|/ 25.0	5
||/	cube root	||/ 27.0	3
!	factorial	5 !	120
!!	factorial (prefix operator)	!! 5	120
@	absolute value	@ -5.0	5
&	bitwise AND	91 & 15	11
|	bitwise OR	32 | 3	35
#	bitwise XOR	17 # 5	20
~	bitwise NOT	~1	-2
<<	bitwise shift left	1 << 4	16
>>	bitwise shift right	8 >> 2	2

- Functions

Function	Return Type	Description	Example	Result
`abs(x)`	(same as input)	absolute value	`abs(-17.4)`	`17.4`
`cbrt(dp)`	`dp`	cube root	`cbrt(27.0)`	`3`
`ceil(dp or numeric)`	(same as input)	nearest integer greater than or equal to argument	`ceil(-42.8)`	`-42`
`ceiling(dp or numeric)`	(same as input)	nearest integer greater than or equal to argument (same as `ceil`)	`ceiling(-95.3)`	`-95`
`degrees(dp)`	`dp`	radians to degrees	`degrees(0.5)`	`28.6478897565412`
`div(y numeric, x numeric)`	`numeric`	integer quotient of *y*/*x*	`div(9,4)`	`2`
`exp(dp or numeric)`	(same as input)	exponential	`exp(1.0)`	`2.71828182845905`
`floor(dp or numeric)`	(same as input)	nearest integer less than or equal to argument	`floor(-42.8)`	`-43`
`ln(dp or numeric)`	(same as input)	natural logarithm	`ln(2.0)`	`0.693147180559945`
`log(dp or numeric)`	(same as input)	base 10 logarithm	`log(100.0)`	`2`
`log10(dp or numeric)`	(same as input)	base 10 logarithm	`log10(100.0)`	`2`
`log(b numeric, x numeric)`	`numeric`	logarithm to base *b*	`log(2.0, 64.0)`	`6.0000000000`
`mod(y, x)`	(same as argument types)	remainder of *y*/*x*	`mod(9,4)`	`1`
`pi()`	`dp`	"π" constant	`pi()`	`3.14159265358979`
`power(a dp, b dp)`	`dp`	*a* raised to the power of *b*	`power(9.0, 3.0)`	`729`
`power(a numeric, b numeric)`	`numeric`	*a* raised to the power of *b*	`power(9.0, 3.0)`	`729`
`radians(dp)`	`dp`	degrees to radians	`radians(45.0)`	`0.785398163397448`
`round(dp or numeric)`	(same as input)	round to nearest integer	`round(42.4)`	`42`
`round(v numeric, s int)`	`numeric`	round to *s* decimal places	`round(42.4382, 2)`	`42.44`
`scale(numeric)`	`integer`	scale of the argument (the number of decimal digits in the fractional part)	`scale(8.41)`	`2`
`sign(dp or numeric)`	(same as input)	sign of the argument (-1, 0, +1)	`sign(-8.4)`	`-1`
`sqrt(dp or numeric)`	(same as input)	square root	`sqrt(2.0)`	`1.4142135623731`
`trunc(dp or numeric)`	(same as input)	truncate toward zero	`trunc(42.8)`	`42`
`trunc(v numeric, s int)`	`numeric`	truncate to *s* decimal places	`trunc(42.4382, 2)`	`42.43`

String Functions and Operations

SELECT upper (first_name) || ‘ ‘ || upper (last_name) AS full_name
FROM customer

	full_name text
1	JARED ELY
2	MARY SMITH
3	PATRICIA JOHNSON
4	LINDA WILLIAMS
5	BARBARA JONES

Function	Return Type	Description	Example	Result
`string || string`	**`text`**	String concatenation	`'Post' || 'greSQL'`	`PostgreSQL`
`string || non-string` or `non-string || string`	**`text`**	String concatenation with one non-string input	`'Value: ' || 42`	`Value: 42`
`bit_length(string)`	**`int`**	Number of bits in string	`bit_length('jose')`	`32`
`char_length(string)` or `character_length(string)`	**`int`**	Number of characters in string	`char_length('jose')`	`4`
`lower(string)`	**`text`**	Convert string to lower case	`lower('TOM')`	`tom`
`octet_length(string)`	**`int`**	Number of bytes in string	`octet_length('jose')`	`4`
`overlay(string placing string from int [for int])`	**`text`**	Replace substring	`overlay('Txxxxas' placing 'hom' from 2 for 4)`	`Thomas`
`position(substring in string)`	**`int`**	Location of specified substring	`position('om' in 'Thomas')`	`3`
`substring(string [from int] [for int])`	**`text`**	Extract substring	`substring('Thomas' from 2 for 3)`	`hom`
`substring(string from` ***`pattern`***`)`	**`text`**	Extract substring matching POSIX regular expression. See Section 9.7 for more information on pattern matching.	`substring('Thomas' from '...$')`	`mas`
`substring(string from` ***`pattern`*** `for` ***`escape`***`)`	**`text`**	Extract substring matching SQL regular expression. See Section 9.7 for more information on pattern matching.	`substring('Thomas' from '%#"o_a#"_' for '#')`	`oma`
`trim([leading | trailing | both] [characters] from string)`	**`text`**	Remove the longest string containing only characters from **`characters`** (a space by default) from the start, end, or both ends (`both` is the default) of **`string`**	`trim(both 'xyz' from 'yxTomxx')`	`Tom`
`trim([leading | trailing | both] [from] string [, characters] )`	**`text`**	Non-standard syntax for `trim()`	`trim(both from 'yxTomxx', 'xyz')`	`Tom`
`upper(string)`	**`text`**	Convert string to upper case	`upper('tom')`	`TOM`

Subquery

- The subquery is performed first since it is inside the parenthesis.
- We can also use the **IN** operator in conjunction with a subquery to check against multiple results returned.

How can we get a list of students who scored better than the average grade?
Firstly, we get the average score:

```
SELECT AVG(grade) FROM test_scores
```

Secondly, put average score under conditional query:

```
SELECT student, grade
FROM test_scores
WHERE grade > (SELECT AVG (grade)
               FROM test_scores)
```

- The **EXISTS** operator is used to test for existence of rows in a subquery.
- Typically, a subquery is passed in the **EXISTS ()** function to check if any rows are returned with the subquery (return T/F).

Typical Syntax:

```
SELECT column_name
FROM table_name
WHERE EXISTS (SELECT column_name
              FROM table_name
              WHERE condition)
```

Find customers who have at least one payment whose amount Is greater than 11 and list the first name and last name of those customers. (Hint: always shows condition first, here is amount > 11)

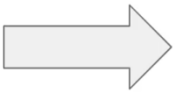
```
SELECT first_name, last_name
FROM customer AS c
WHERE EXISTS (SELECT * FROM payment AS p
              WHERE p.customer_id = c.customer_id
              AND amount > 11)
```

Self-Join

- A Self-Join is a query in which a table joined to itself.
- Self-Joins are useful for comparing values in a column of rows within the same table.
- When using a Self-Join, it is necessary to use an alias for the table, otherwise the table names would be ambiguous.

Syntax:
SELECT tableA.col, tableB.col
FROM table **AS** tableA
JOIN table **AS** tableB
ON tableA.some_col = tableB.other_col

EMPLOYEES		
emp_id	name	report_id
1	Andrew	3
2	Bob	3
3	Charlie	4
4	David	1



name	rep
Andrew	Charlie
Bob	Charlie
Charlie	David
David	Andrew

Step 1: the main table is “employees”, so replace table to “employees”
SELECT tableA.col, tableB.col
FROM employees **AS** tableA
JOIN employees **AS** tableB
ON tableA.some_col = tableB.other_col

Step 2: tableA should be “emp”, and table B should be “report”
SELECT emp.col, report.col
FROM employees **AS** emp
JOIN employees **AS** report
ON emp.some_col = report.other_col

Step 3: what we care about are “emp_id” (in “emp” table) and “report_id” (in “report” table)
SELECT emp.col, report.col
FROM employees **AS** emp
JOIN employees **AS** report
ON emp.emp_id = report.report_id

Step 4: what we need is their names
SELECT emp.name, report.name

FROM employees **AS** emp
JOIN employees **AS** report
ON emp.emp_id = report.report_id

Ex: Find all the pairs of films that have the same length
My answer:
SELECT f1.title, f2.title
FROM film **AS** f1
JOIN film **AS** f2
ON f1.length = f2.length

	title character varying (255)	title character varying (255)
1	Chamber Italian	Resurrection Silverado
2	Chamber Italian	Magic Mallrats
3	Chamber Italian	Graffiti Love
4	Chamber Italian	Affair Prejudice
5	Chamber Italian	Chamber Italian
6	Grosse Wonderful	Hurricane Affair
7	Grosse Wonderful	Hook Chariots

The 5th row got the same pair, so we need to remove this row.
Correct answer:
SELECT f1.title, f2.title
FROM film AS f1
JOIN film AS f2
ON f1.film_id != f2.film_id
AND f1.length = f2.length

	title character varying (255)	title character varying (255)
1	Chamber Italian	Resurrection Silverado
2	Chamber Italian	Magic Mallrats
3	Chamber Italian	Graffiti Love
4	Chamber Italian	Affair Prejudice
5	Grosse Wonderful	Hurricane Affair
6	Grosse Wonderful	Hook Chariots

Assessment Test 2

1. How can you retrieve all the information from the cd.facilities table?

   SELECT * FROM cd.facilities

	facid [PK] integer	name character varying (100)	membercost numeric	guestcost numeric	initialoutlay numeric	monthlymaintenance numeric
1	0	Tennis Court 1	5	25	10000	200
2	1	Tennis Court 2	5	25	8000	200
3	2	Badminton Court	0	15.5	4000	50
4	3	Table Tennis	0	5	320	10
5	4	Massage Room 1	35	80	4000	3000

2. You want to print out a list of all of the facilities and their cost to members. How would you retrieve a list of only facility names and costs?

   SELECT facilities.name, membercost FROM cd.facilities

	name character varying (100)	membercost numeric
1	Tennis Court 1	5
2	Tennis Court 2	5
3	Badminton Court	0
4	Table Tennis	0
5	Massage Room 1	35

3. How can you produce a list of facilities that charge a fee to members?

   SELECT * FROM cd.facilities
   WHERE membercost > 0

	facid [PK] integer	name character varying (100)	membercost numeric	guestcost numeric	initialoutlay numeric	monthlymaintenance numeric
1	0	Tennis Court 1	5	25	10000	200
2	1	Tennis Court 2	5	25	8000	200
3	4	Massage Room 1	35	80	4000	3000
4	5	Massage Room 2	35	80	4000	3000
5	6	Squash Court	3.5	17.5	5000	80

4. 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, facilities.name, membercost, monthlymaintenance
   FROM cd.facilities
   WHERE membercost > 0
   AND membercost < monthlymaintenance * 1/50.0

	facid [PK] integer	name character varying (100)	membercost numeric	monthlymaintenance numeric
1	4	Massage Room 1	35	3000
2	5	Massage Room 2	35	3000

5. How can you produce a list of all facilities with the word 'Tennis' in their name?

   SELECT * FROM cd.facilities
   WHERE facilities.name LIKE '%Tennis%' (ILIKE: ignore case)

   %: in a pattern matches any sequence of zero or more characters
   _: in a pattern matches any single character

	facid [PK] integer	name character varying (100)	membercost numeric	guestcost numeric	initialoutlay numeric	monthlymaintenance numeric
1	0	Tennis Court 1	5	25	10000	200
2	1	Tennis Court 2	5	25	8000	200
3	3	Table Tennis	0	5	320	10

6. 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)

	facid [PK] integer	name character varying (100)	membercost numeric	guestcost numeric	initialoutlay numeric	monthlymaintenance numeric
1	1	Tennis Court 2	5	25	8000	200
2	5	Massage Room 2	35	80	4000	3000

7. 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 > '2012-08-31'

	memid [PK] integer	surname character varying (200)	firstname character varying (200)	joindate timestamp without time zone
1	24	Sarwin	Ramnaresh	2012-09-01 08:44:42
2	26	Jones	Douglas	2012-09-02 18:43:05
3	27	Rumney	Henrietta	2012-09-05 08:42:35
4	28	Farrell	David	2012-09-15 08:22:05
5	29	Worthington-Smyth	Henry	2012-09-17 12:27:15

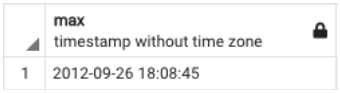
8. How can you produce an ordered list of the first 10 surnames in the members table? The list must not contain duplicates.

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

	surname character varying (200)
1	Bader
2	Baker
3	Boothe
4	Butters

9. You'd like to get the signup date of your last member. How can you retrieve this information? (Hint: the maximum date)

   SELECT MAX (joindate) FROM cd.members



	max timestamp without time zone
1	2012-09-26 18:08:45