DATE_TRUNC

TIMESTAMP Format:

YYYY-MM-DD HH:MI:SS

DATE_TRUNC allows you to truncate your date to a particular part of your date-time column.

General Form:

DATE_TRUNC('[unit of time]', time_attribute)

[unit of time] refers to the desired unit of time at which you want to aggregate records.

Time_attribute refers to the column that contains the timestamp that you would like to round.

We can round off a timestamp to the ff.units of time:

-microsecond
-millisecond
-second
-minute
-hour
-day
- week
-month
- quarter
- year
-decade
-century
-millenium

Example1:

Show the monthly revenue per store. Hint: You need to show the revenue by month and by store.

Query Editor | Query History

```
1  SELECT DATE_TRUNC('month', payment_date) AS month,
2      SUM (amount) AS monthly_revenue, store_id
3  FROM customer c
4  JOIN payment p
5  ON c.customer_id = p.customer_id
6  GROUP BY 1,3
```

Data Output | Explain | Messages | Notifications

	month timestamp without time zone	monthly_revenue numeric	store_id smallint
1	2007-04-01 00:00:00	12701.47	2
2	2007-05-01 00:00:00	231.16	2
3	2007-03-01 00:00:00	13017.32	1
4	2007-02-01 00:00:00	3888.75	2
5	2007-03-01 00:00:00	10869.24	2
6	2007-05-01 00:00:00	283.02	1
7	2007-02-01 00:00:00	4463.09	1
8	2007-04-01 00:00:00	15857.99	1

Example2:

How many distinct customers doe the company have per month? What is the average number of rentals per customer?

Hint: The average number of rentals per customer can be obtained by dividing the number of rentals by the number of unique customers.

Query Editor | Query History

```sql
SELECT DATE_TRUNC('month', rental_date) AS month,
    COUNT(rental_id) AS total_rentals,
    COUNT(DISTINCT(customer_id)) AS unique_customers,
    COUNT(rental_id)/COUNT(DISTINCT(customer_id)) AS avg_num_per_customer
FROM rental
GROUP BY 1;
```

Data Output | Explain | Messages | Notifications

	month timestamp without time zone	total_rentals bigint	unique_customers bigint	avg_num_per_customer bigint
1	2005-05-01 00:00:00	1156	520	2
2	2005-06-01 00:00:00	2311	590	3
3	2005-07-01 00:00:00	6709	599	11
4	2005-08-01 00:00:00	5686	599	9
5	2006-02-01 00:00:00	182	158	1

TRY IT

1. How many unique films are rented each month? Show the month and number of unique films rented out each month. Rename the attributes accordingly.

Hint: You may join the inventory table (which contains film ID) to the rental table.

```sql
SELECT DATE_TRUNC('month', rental_date),
        COUNT(DISTINCT (i.film_id)) AS num_unique_films
FROM inventory i
JOIN rental r
USING (inventory_id)
GROUP BY 1;
```

2. Show the average amount paid per customer in each month.

The average amount paid per customer is equal to the total amount paid divided by the number of **distinct** customers. Rename the attribute as "Average Amount Paid per Customer"

Hint: All the information you need is available in the payment table.

```sql
SELECT DATE_TRUNC('month', payment_date),
        SUM(amount) AS total_amount,
        COUNT(DISTINCT(customer_id)) AS unique_customers,
      ROUND(SUM(amount) / COUNT(DISTINCT customer_id),2)
            AS "Average Amount Paid Per Customer"
FROM payment
GROUP BY 1;
```

HAVING

HAVING is usually used in conjunction with GROUP BY to filter the results of an aggregation.
WHERE is a condition on individual rows while HAVING is a condition on the results of an aggregation.

General Form:

```
SELECT column1, aggregate_function(attribute)
FROM table1
GROUP BY column1
HAVING aggregate_function(attribute) > 10
```

ORDER OF STATEMENTS

SELECT > FROM > WHERE > GROUP BY > HAVING > ORDER BY > LIMIT

Example 1:

Let's define power customer as the customers who brought in at least $200 to the company. Provide a list of the company's power customers.
Show their customer ID first name, last name and total amount paid to the company.

Query Editor | Query History

```
1  SELECT c.customer_id, first_name, last_name, SUM(amount) AS sum_amount
2  FROM customer c
3  JOIN payment p
4  ON c.customer_id = p.customer_id
5  GROUP BY 1
6  HAVING SUM(amount) >= 200
7  ORDER BY 4 DESC;
```

Data Output | Explain | Messages | Notifications

	customer_id [PK] integer	first_name character varying (45)	last_name character varying (45)	sum_amount numeric
1	148	Eleanor	Hunt	211.55
2	526	Karl	Seal	208.58

Example 2:

Show the average rental_rate (rounded up to 2 decimal places) for each of the following film categories: children, action, animation, comedy and family. In particular, only show those categories with average rental rate less than $3 (The table must show the category id, category name(renamed to category) and the average rental rate.

Query Editor | Query History

```
1  SELECT c.category_id, name AS categaory,
2      ROUND(AVG(rental_rate), 2) AS avg_rental_rate
3  FROM film f
4  JOIN film_category fc
5  ON f.film_id = fc.film_id
6  JOIN category c
7  ON fc.category_id = c.category_id
8  WHERE c.category_id IN (1,2,3,5,8)
9  GROUP BY 1
10 HAVING AVG (rental_rate) < 3;
```

Data Output | Explain | Messages | Notifications

	category_id [PK] integer	categaory character varying (25)	avg_rental_rate numeric
1	1	Action	2.65
2	3	Children	2.89
3	2	Animation	2.81
4	8	Family	2.76

Question:
1.

The marketing team wants to identify customers who have rented films at least 30 times (also called the reward customers). These customers are eligible to receive 2 FREE film rentals.

Help the marketing team identify these reward customers. Provide a table showing the customer ID, first name, last name and the cumulative number of rentals for each customer (with the highest number of rentals on top).

```
SELECT c.customer_id, first_name, last_name,
        COUNT(rental_id) AS rental_count
FROM customer c
JOIN rental r
ON c.customer_id = r.customer_id
GROUP BY 1
HAVING COUNT(rental_id) >= 30
ORDER BY 1;
```

The COO wants to send a friendly warning to those customers who, on average, return films after 5 days (past due). Show the customer ID of customers with average rental duration of more than 5 days. Sort the results by customer ID.

Hint: You may find the rental duration in the film table. The nearest table to the film table with customer ID is rental table.

```
SELECT customer_id,
   ROUND(AVG(rental_duration),2) AS "Average Rental
                                    Duration"
FROM film f
JOIN inventory i
ON f.film_id = i.film_id
JOIN rental r
ON i.inventory_id = r.inventory_id
GROUP BY 1
HAVING ROUND(AVG(rental_duration),2) > 5
ORDER BY 1;
```