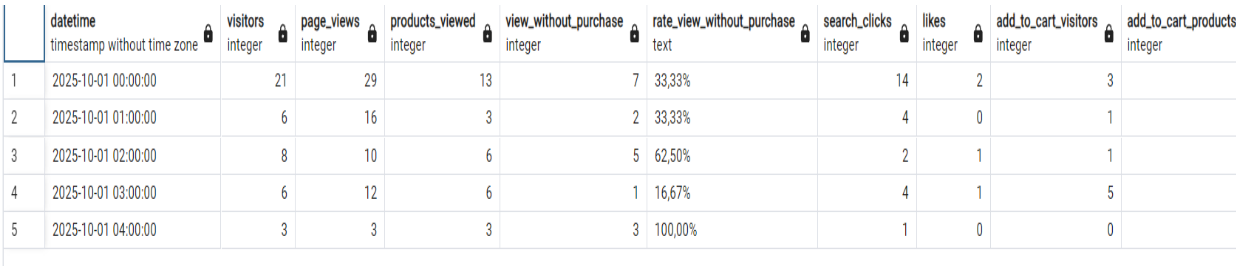
SQL for Data Analysis

Objective: Use SQL queries to extract and analyze data from a database.
Tool: PostgreSQL
Queries:

-- BASIC SQL QUERIES (SELECT, WHERE, ORDER BY, GROUP BY)

-- 1. Select all data

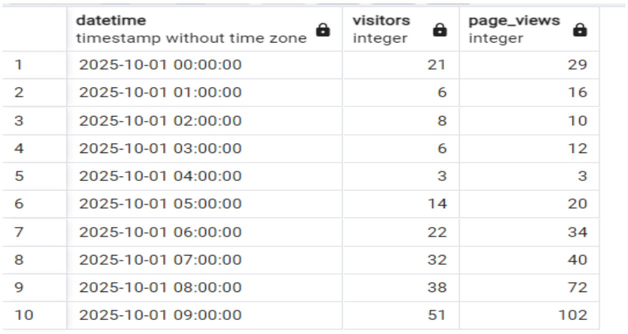
SELECT * FROM ecommerce_hourly LIMIT 5;



	datetime timestamp without time zone	visitors integer	page_views integer	products_viewed integer	view_without_purchase integer	rate_view_without_purchase text	search_clicks integer	likes integer	add_to_cart_visitors integer	add_to_cart_products integer
1	2025-10-01 00:00:00	21	29	13	7	33,33%	14	2	3	
2	2025-10-01 01:00:00	6	16	3	2	33,33%	4	0	1	
3	2025-10-01 02:00:00	8	10	6	5	62,50%	2	1	1	
4	2025-10-01 03:00:00	6	12	6	1	16,67%	4	1	5	
5	2025-10-01 04:00:00	3	3	3	3	100,00%	1	0	0	

-- 2. Select specific ccolumns

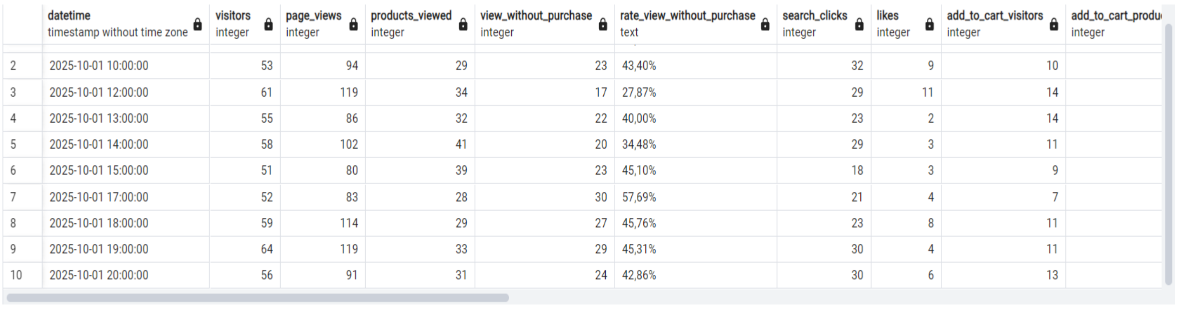
SELECT datetime, visitors, page_views
FROM ecommerce_hourly;



	datetime timestamp without time zone	visitors integer	page_views integer
1	2025-10-01 00:00:00	21	29
2	2025-10-01 01:00:00	6	16
3	2025-10-01 02:00:00	8	10
4	2025-10-01 03:00:00	6	12
5	2025-10-01 04:00:00	3	3
6	2025-10-01 05:00:00	14	20
7	2025-10-01 06:00:00	22	34
8	2025-10-01 07:00:00	32	40
9	2025-10-01 08:00:00	38	72
10	2025-10-01 09:00:00	51	102

-- 3. Filter rows (Hours where visitor > 50)

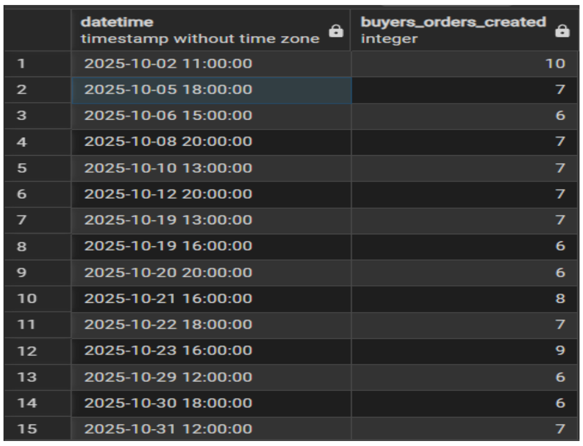
SELECT * FROM ecommerce_hourly
WHERE visitors > 50 LIMIT 10;



	datetime timestamp without time zone	visitors integer	page_views integer	products_viewed integer	view_without_purchase integer	rate_view_without_purchase text	search_clicks integer	likes integer	add_to_cart_visitors integer	add_to_cart_produ integer
2	2025-10-01 10:00:00	53	94	29	23	43,40%	32	9	10	
3	2025-10-01 12:00:00	61	119	34	17	27,87%	29	11	14	
4	2025-10-01 13:00:00	55	86	32	22	40,00%	23	2	14	
5	2025-10-01 14:00:00	58	102	41	20	34,48%	29	3	11	
6	2025-10-01 15:00:00	51	80	39	23	45,10%	18	3	9	
7	2025-10-01 17:00:00	52	83	28	30	57,69%	21	4	7	
8	2025-10-01 18:00:00	59	114	29	27	45,76%	23	8	11	
9	2025-10-01 19:00:00	64	119	33	29	45,31%	30	4	11	
10	2025-10-01 20:00:00	56	91	31	24	42,86%	30	6	13	

-- 4. Hours with more than 5 orders created

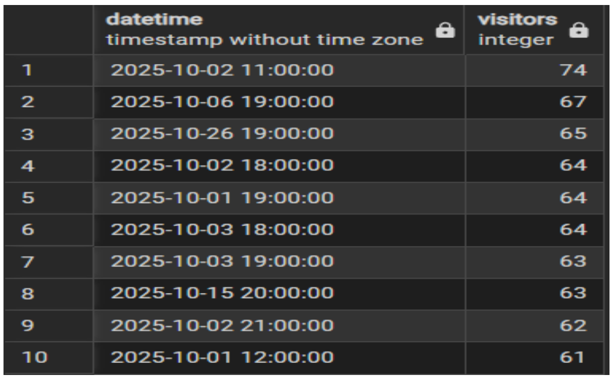
SELECT datetime, buyers_orders_created FROM ecommerce_hourly
WHERE buyers_orders_created > 5;



	datetime timestamp without time zone	buyers_orders_created integer
1	2025-10-02 11:00:00	10
2	2025-10-05 18:00:00	7
3	2025-10-06 15:00:00	6
4	2025-10-08 20:00:00	7
5	2025-10-10 13:00:00	7
6	2025-10-12 20:00:00	7
7	2025-10-19 13:00:00	7
8	2025-10-19 16:00:00	6
9	2025-10-20 20:00:00	6
10	2025-10-21 16:00:00	8
11	2025-10-22 18:00:00	7
12	2025-10-23 16:00:00	9
13	2025-10-29 12:00:00	6
14	2025-10-30 18:00:00	6
15	2025-10-31 12:00:00	7

-- 5. Sort by datetime and visitors

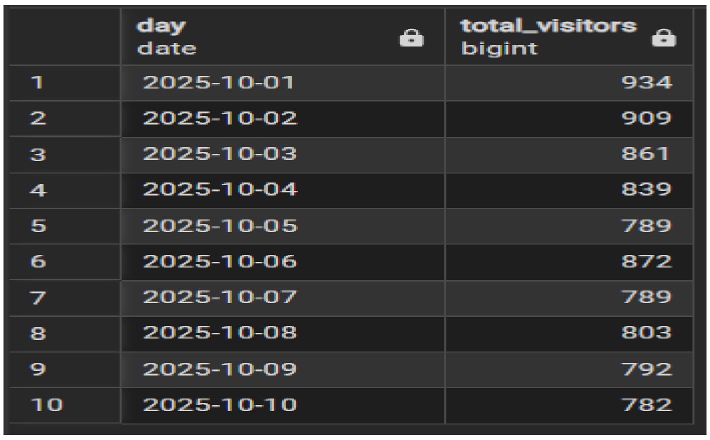
```
SELECT datetime, visitors FROM ecommerce_hourly
ORDER BY visitors DESC LIMIT 10;
```



	datetime timestamp without time zone	visitors integer
1	2025-10-02 11:00:00	74
2	2025-10-06 19:00:00	67
3	2025-10-26 19:00:00	65
4	2025-10-02 18:00:00	64
5	2025-10-01 19:00:00	64
6	2025-10-03 18:00:00	64
7	2025-10-03 19:00:00	63
8	2025-10-15 20:00:00	63
9	2025-10-02 21:00:00	62
10	2025-10-01 12:00:00	61

-- 6. Group by day -> total visitors per day

```
SELECT DATE(datetime) AS day, SUM(visitors) AS total_visitors FROM ecommerce_hourly
GROUP BY DATE(datetime)
ORDER BY day LIMIT 10;
```

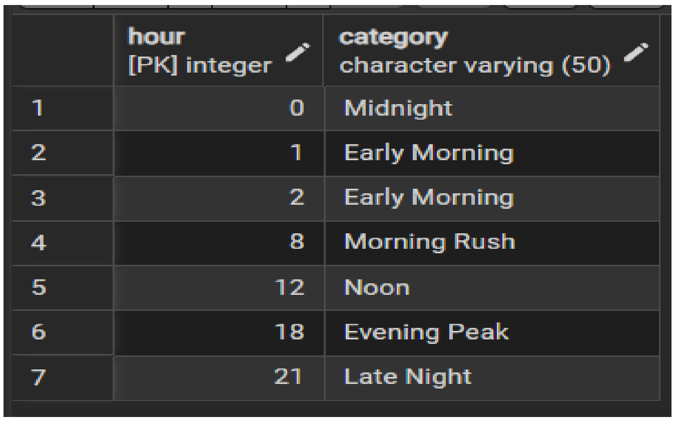


	day date	total_visitors bigint
1	2025-10-01	934
2	2025-10-02	909
3	2025-10-03	861
4	2025-10-04	839
5	2025-10-05	789
6	2025-10-06	872
7	2025-10-07	789
8	2025-10-08	803
9	2025-10-09	792
10	2025-10-10	782

-- Queries (INNER, RIGHT, LEFT)

-- 1. Create a Basic table for example

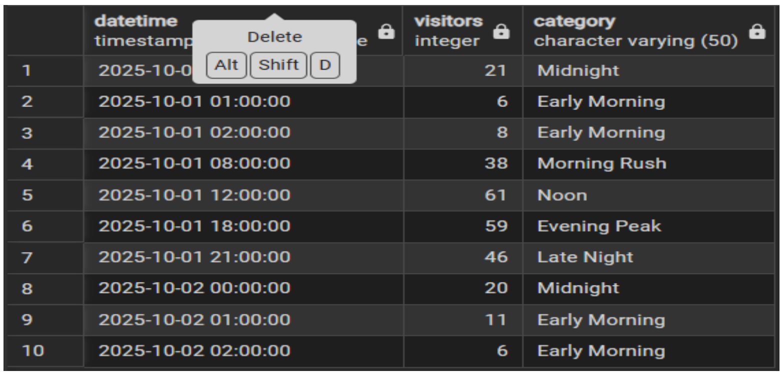
```
CREATE TABLE hour_category (
   hour INT PRIMARY KEY,
   category VARCHAR(50));
INSERT INTO hour_category VALUES
(0, 'Midnight'),
(1, 'Early Morning'),
(2, 'Early Morning'),
(8, 'Morning Rush'),
(12, 'Noon'),
(18, 'Evening Peak'),
(21, 'Late Night');
SELECT * FROM hour_category;
```



	hour [PK] integer	category character varying (50)
1	0	Midnight
2	1	Early Morning
3	2	Early Morning
4	8	Morning Rush
5	12	Noon
6	18	Evening Peak
7	21	Late Night

-- 2. Inner Join (Match Hour with Category)

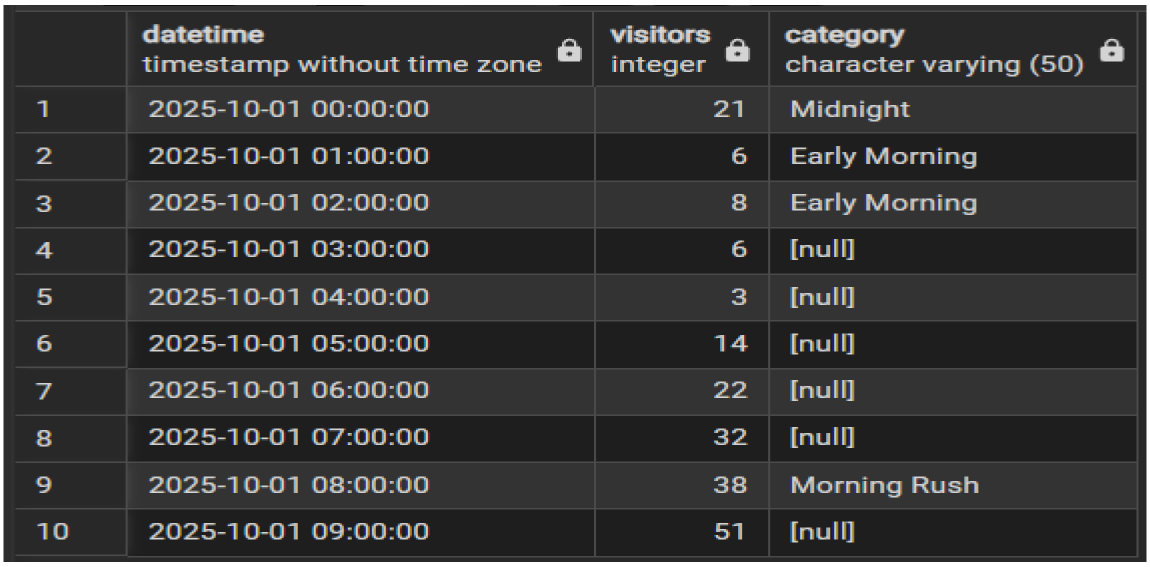
```
SELECT e.datetime, e.visitors, c.category FROM ecommerce_hourly e
INNER JOIN hour_category c
ON EXTRACT(HOUR FROM e.datetime) = c.hour LIMIT 10;
```



	datetime timestamp	visitors integer	category character varying (50)
1	2025-10-0	21	Midnight
2	2025-10-01 01:00:00	6	Early Morning
3	2025-10-01 02:00:00	8	Early Morning
4	2025-10-01 08:00:00	38	Morning Rush
5	2025-10-01 12:00:00	61	Noon
6	2025-10-01 18:00:00	59	Evening Peak
7	2025-10-01 21:00:00	46	Late Night
8	2025-10-02 00:00:00	20	Midnight
9	2025-10-02 01:00:00	11	Early Morning
10	2025-10-02 02:00:00	6	Early Morning

-- 3. Left Join (Show All data even if category missing)

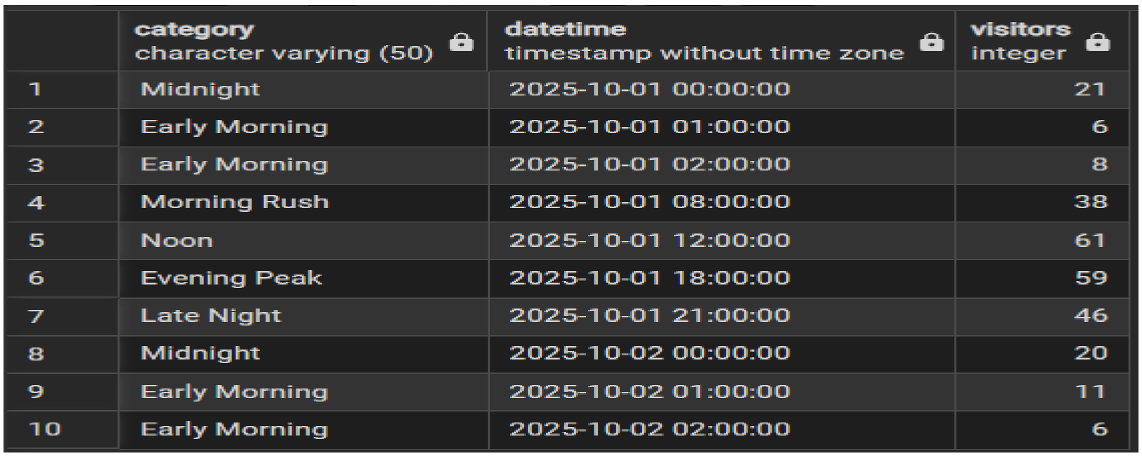
```
SELECT e.datetime, e.visitors, c.category
FROM ecommerce_hourly e
LEFT JOIN hour_category c
ON EXTRACT(HOUR FROM e.datetime) = c.hour LIMIT 10;
```



	datetime timestamp without time zone	visitors integer	category character varying (50)
1	2025-10-01 00:00:00	21	Midnight
2	2025-10-01 01:00:00	6	Early Morning
3	2025-10-01 02:00:00	8	Early Morning
4	2025-10-01 03:00:00	6	[null]
5	2025-10-01 04:00:00	3	[null]
6	2025-10-01 05:00:00	14	[null]
7	2025-10-01 06:00:00	22	[null]
8	2025-10-01 07:00:00	32	[null]
9	2025-10-01 08:00:00	38	Morning Rush
10	2025-10-01 09:00:00	51	[null]

-- 4. Right Join (Show All Categories even if no matching data)

```
SELECT c.category, e.datetime, e.visitors FROM ecommerce_hourly e
RIGHT JOIN hour_category c
ON EXTRACT(HOUR FROM e.datetime) = c.hour LIMIT 10;
```

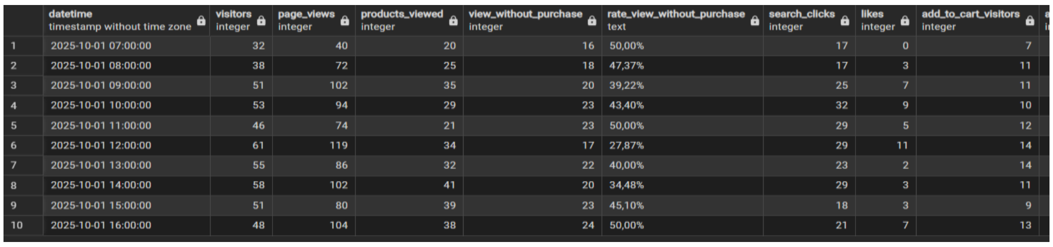


	category character varying (50)	datetime timestamp without time zone	visitors integer
1	Midnight	2025-10-01 00:00:00	21
2	Early Morning	2025-10-01 01:00:00	6
3	Early Morning	2025-10-01 02:00:00	8
4	Morning Rush	2025-10-01 08:00:00	38
5	Noon	2025-10-01 12:00:00	61
6	Evening Peak	2025-10-01 18:00:00	59
7	Late Night	2025-10-01 21:00:00	46
8	Midnight	2025-10-02 00:00:00	20
9	Early Morning	2025-10-02 01:00:00	11
10	Early Morning	2025-10-02 02:00:00	6

-- Sub Queries (Nested Queries)

-- 1. Find rows where visitors > average visitors

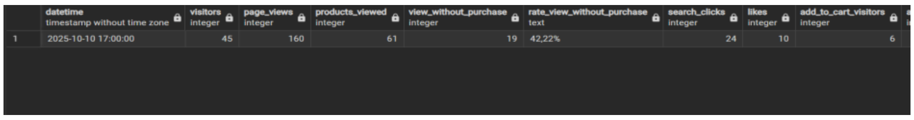
```
SELECT * FROM ecommerce_hourly
WHERE visitors >
   (SELECT AVG(visitors) FROM ecommerce_hourly) LIMIT 10;
```



	datetime timestamp without time zone	visitors integer	page_views integer	products_viewed integer	view_without_purchase integer	rate_view_without_purchase text	search_clicks integer	likes integer	add_to_cart_visitors integer
1	2025-10-01 07:00:00	32	40	20	16	50,00%	17	0	7
2	2025-10-01 08:00:00	38	72	25	18	47,37%	17	3	11
3	2025-10-01 09:00:00	51	102	35	20	39,22%	25	7	11
4	2025-10-01 10:00:00	53	94	29	23	43,40%	32	9	10
5	2025-10-01 11:00:00	46	74	21	23	50,00%	29	5	12
6	2025-10-01 12:00:00	61	119	34	17	27,87%	29	11	14
7	2025-10-01 13:00:00	55	86	32	22	40,00%	23	2	14
8	2025-10-01 14:00:00	58	102	41	20	34,48%	29	3	11
9	2025-10-01 15:00:00	51	80	39	23	45,10%	18	3	9
10	2025-10-01 16:00:00	48	104	38	24	50,00%	21	7	13

-- 2. Highest product views using subquery

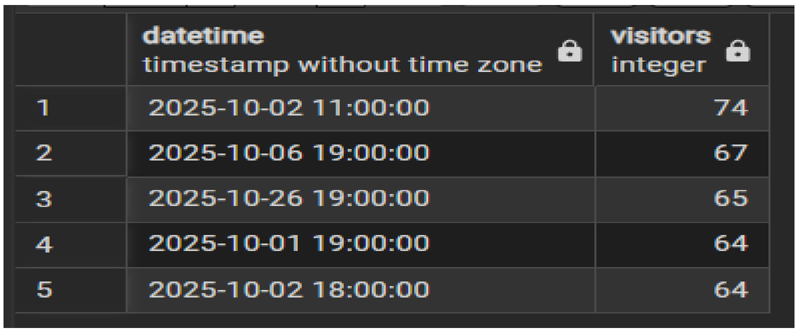
```
SELECT * FROM ecommerce_hourly
WHERE products_viewed = (
   SELECT MAX(products_viewed) FROM ecommerce_hourly) LIMIT 10;
```



	datetime timestamp without time zone	visitors integer	page_views integer	products_viewed integer	view_without_purchase integer	rate_view_without_purchase text	search_clicks integer	likes integer	add_to_cart_visitors integer
1	2025-10-10 17:00:00	45	160	61	19	42,22%	24	10	6

-- 3. Top 5 busiest hours using subquery

```
SELECT datetime, visitors FROM ecommerce_hourly
ORDER BY visitors DESC LIMIT 5;
```

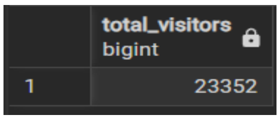


	datetime timestamp without time zone	visitors integer
1	2025-10-02 11:00:00	74
2	2025-10-06 19:00:00	67
3	2025-10-26 19:00:00	65
4	2025-10-01 19:00:00	64
5	2025-10-02 18:00:00	64

-- Aggregate Functions (SUM, AVG, MAX, MIN)

-- 1. Total Visitors

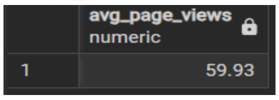
```
SELECT SUM(visitors) AS total_visitors FROM ecommerce_hourly;
```



	total_visitors bigint
1	23352

-- 2. Average Page Views

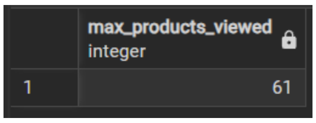
```
SELECT ROUND(AVG
(page_views), 2) AS avg_page_views FROM ecommerce_hourly;
```



	avg_page_views numeric
1	59.93

-- 3. Max product viewed in an hour

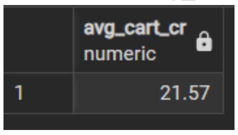
```
SELECT MAX(products_viewed) AS max_products_viewed FROM ecommerce_hourly;
```



	max_products_viewed integer
1	61

-- 4. Average add to cart conversion rate

```
SELECT ROUND(AVG(REPLACE(REPLACE(cr_products_added_to_cart, '%', ''),',','.')::NUMERIC
       ),2) AS avg_cart_cr FROM ecommerce_hourly;
```

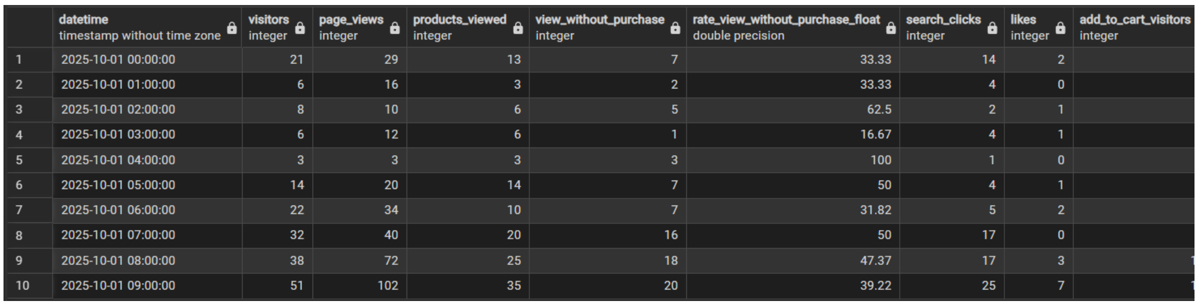


	avg_cart_cr numeric
1	21.57

-- Create View for Analysis

-- View → Cleaned dataset with numeric conversion

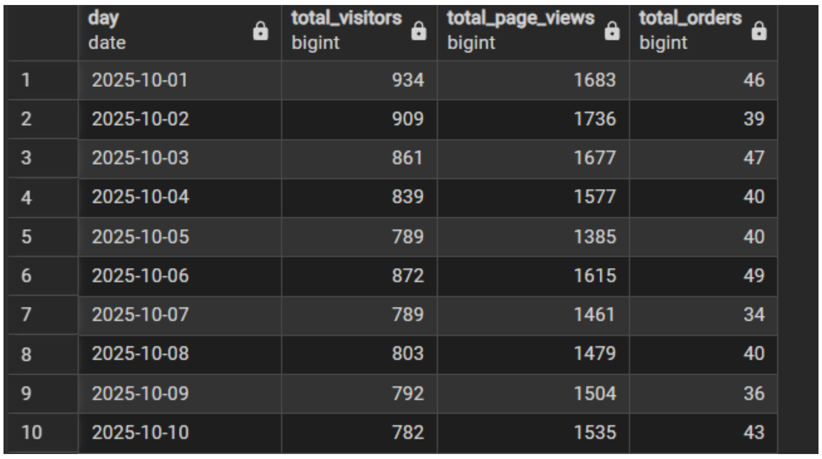
```
CREATE OR REPLACE VIEW ecommerce_cleaned AS
SELECT
   datetime, visitors, page_views, products_viewed, view_without_purchase,
   REPLACE(REPLACE(rate_view_without_purchase, '%', ''), ',', '.')::FLOAT AS
rate_view_without_purchase_float,
   search_clicks, likes, add_to_cart_visitors, add_to_cart_products,
   REPLACE(REPLACE(cr_products_added_to_cart, '%', ''), ',', '.')::FLOAT AS cr_products_added_float,
   buyers_orders_created,
   products_orders_created,
   products_ordered,
   REPLACE(REPLACE(cr_orders_created, '%', ''), ',', '.')::FLOAT AS cr_orders_created_float,
   buyers_ready_to_ship,
   products_ready_to_ship_original,
   products_ready_to_ship,
   REPLACE(REPLACE(cr_ready_to_ship, '%', ''), ',', '.')::FLOAT AS cr_ready_to_ship_float,
   REPLACE(REPLACE(cr_ready_to_ship_over_orders_created, '%', ''), ',', '.')::FLOAT AS
cr_ready_ship_over_orders_float
FROM ecommerce_hourly;

SELECT * FROM ecommerce_cleaned LIMIT 10;
```



	datetime timestamp without time zone	visitors integer	page_views integer	products_viewed integer	view_without_purchase integer	rate_view_without_purchase_float double precision	search_clicks integer	likes integer	add_to_cart_visitors integer
1	2025-10-01 00:00:00	21	29	13	7	33.33	14	2	
2	2025-10-01 01:00:00	6	16	3	2	33.33	4	0	
3	2025-10-01 02:00:00	8	10	6	5	62.5	2	1	
4	2025-10-01 03:00:00	6	12	6	1	16.67	4	1	
5	2025-10-01 04:00:00	3	3	3	3	100	1	0	
6	2025-10-01 05:00:00	14	20	14	7	50	4	1	
7	2025-10-01 06:00:00	22	34	10	7	31.82	5	2	
8	2025-10-01 07:00:00	32	40	20	16	50	17	0	
9	2025-10-01 08:00:00	38	72	25	18	47.37	17	3	
10	2025-10-01 09:00:00	51	102	35	20	39.22	25	7	

-- View Daily Summary

```
CREATE OR REPLACE VIEW daily_summary AS
SELECT
   DATE(datetime) AS day,
   SUM(visitors) AS total_visitors,
   SUM(page_views) AS total_page_views,
   SUM(buyers_orders_created) AS total_orders
FROM ecommerce_hourly
GROUP BY DATE(datetime)
ORDER BY day;
SELECT * FROM daily_summary LIMIT 10;
```



	day date	total_visitors bigint	total_page_views bigint	total_orders bigint
1	2025-10-01	934	1683	46
2	2025-10-02	909	1736	39
3	2025-10-03	861	1677	47
4	2025-10-04	839	1577	40
5	2025-10-05	789	1385	40
6	2025-10-06	872	1615	49
7	2025-10-07	789	1461	34
8	2025-10-08	803	1479	40
9	2025-10-09	792	1504	36
10	2025-10-10	782	1535	43

-- Performance Optimization Using Indexes

-- 1. Index for faster date filtering

CREATE INDEX idx_datetime ON ecommerce_hourly (datetime);

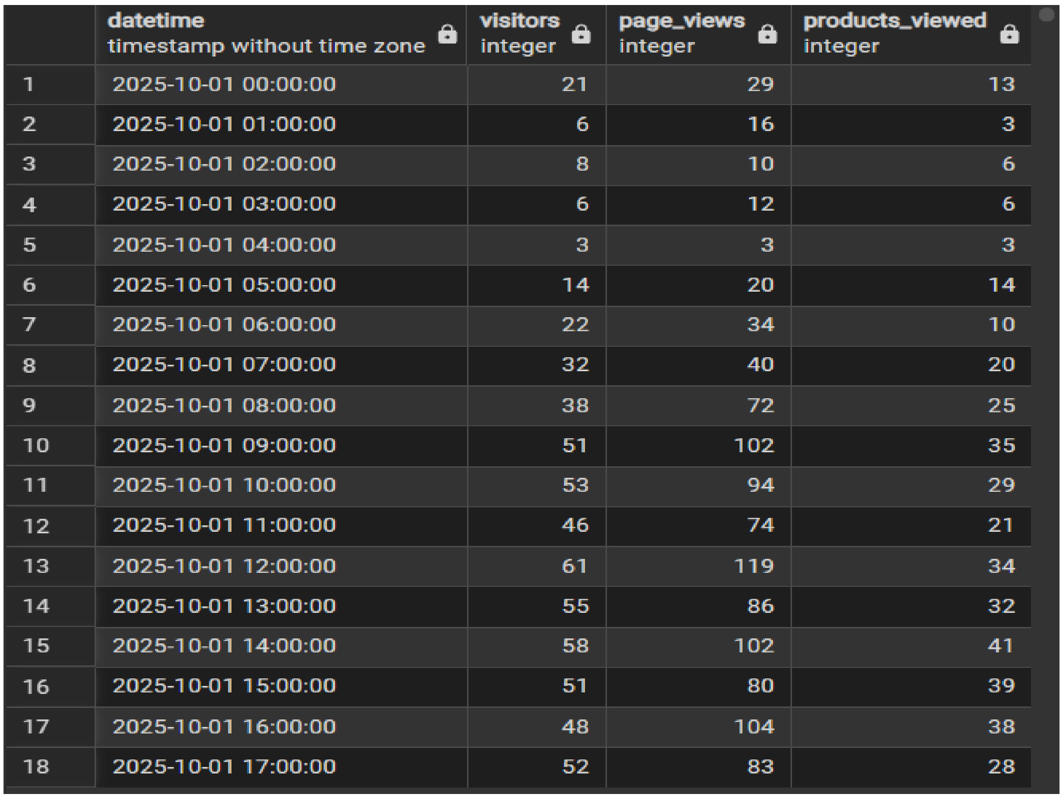
-- 2. Index for visitors (used in WHERE + ORDER BY)

CREATE INDEX idx_visitors ON ecommerce_hourly (visitors);

-- 3. Index for page_views

CREATE INDEX idx_page_views ON ecommerce_hourly (page_views);

SELECT *FROM ecommerce_hourly;



	datetime timestamp without time zone	visitors integer	page_views integer	products_viewed integer
1	2025-10-01 00:00:00	21	29	13
2	2025-10-01 01:00:00	6	16	3
3	2025-10-01 02:00:00	8	10	6
4	2025-10-01 03:00:00	6	12	6
5	2025-10-01 04:00:00	3	3	3
6	2025-10-01 05:00:00	14	20	14
7	2025-10-01 06:00:00	22	34	10
8	2025-10-01 07:00:00	32	40	20
9	2025-10-01 08:00:00	38	72	25
10	2025-10-01 09:00:00	51	102	35
11	2025-10-01 10:00:00	53	94	29
12	2025-10-01 11:00:00	46	74	21
13	2025-10-01 12:00:00	61	119	34
14	2025-10-01 13:00:00	55	86	32
15	2025-10-01 14:00:00	58	102	41
16	2025-10-01 15:00:00	51	80	39
17	2025-10-01 16:00:00	48	104	38
18	2025-10-01 17:00:00	52	83	28