```
CREATE TABLE customers (
    CustomerID VARCHAR(20) PRIMARY KEY,
    CustomerName VARCHAR(100),
    Region VARCHAR(50),
    JoinDate DATE
);

select * from customers;
```

Data Output | Messages | Notifications

Showing rows: 1 to 100

	customerid [PK] character varying (20)	customername character varying (100)	region character varying (50)	joindate date
1	CU1001	Customer_1	North	2020-08-22
2	CU1002	Customer_2	East	2023-12-03
3	CU1003	Customer_3	North	2023-08-10
4	CU1004	Customer_4	North	2022-11-09
5	CU1005	Customer_5	East	2020-03-18
6	CU1006	Customer_6	South	2022-07-23
7	CU1007	Customer_7	South	2023-05-26
8	CU1008	Customer_8	North	2022-04-23
9	CU1009	Customer_9	East	2020-01-09
10	CU1010	Customer_10	East	2024-05-12

Total rows: 100 | Query complete 00:00:00.202

```
CREATE TABLE policies (
    PolicyNumber TEXT PRIMARY KEY,
    CustomerID TEXT,
    StartDate DATE,
    EndDate DATE,
    PremiumAmount INTEGER,
    PolicyType TEXT CHECK (PolicyType IN ('Comprehensive', 'Third-Party', 'Health'))
);

select * from policies;
```

Data Output | Messages | Notifications

Showing rows: 1 to 100

	policynumber [PK] text	customerid text	startdate date	enddate date	premiumamount integer	policytype text
1	P1001	CU1001	2023-04-18	2025-03-15	20199	Comprehensive
2	P1002	CU1002	2023-03-03	2026-02-27	20603	Comprehensive
3	P1003	CU1003	2023-07-15	2024-12-30	31762	Third-Party
4	P1004	CU1004	2023-11-28	2026-12-13	21596	Third-Party
5	P1005	CU1005	2024-04-29	2025-07-03	29717	Third-Party
6	P1006	CU1006	2024-01-01	2025-08-25	29337	Health
7	P1007	CU1007	2023-07-27	2024-10-07	38666	Comprehensive
8	P1008	CU1008	2024-06-28	2026-08-06	48928	Health
9	P1009	CU1009	2024-09-04	2024-11-08	20289	Health
10	P1010	CU1010	2023-01-14	2024-02-19	23190	Health

Total rows: 100 Query complete 00:00:00.243

```
CREATE TABLE claims (
    ClaimID TEXT PRIMARY KEY,
    PolicyNumber TEXT,
    ClaimAmount INTEGER,
    ClaimType TEXT CHECK (ClaimType IN ('Accident', 'Theft', 'Health', 'Fire')),
    Timestamp TIMESTAMP,
    PriorityFlag TEXT CHECK (PriorityFlag IN ('URGENT', 'NORMAL'))
);
```

```
select * from claims;
```

Data Output | Messages | Notifications

Showing rows: 1 to 100 Page No

	claimid [PK] text	policynumber text	claimamount integer	claimtype text	timestamp timestamp without time zone	priorityflag text
1	C1001	P1001	20036	Fire	2025-07-28 07:13:47	NORMAL
2	C1002	P1002	13228	Accident	2025-07-18 03:49:42	URGENT
3	C1003	P1003	70344	Fire	2025-01-22 13:53:42	NORMAL
4	C1004	P1004	38606	Accident	2025-08-08 09:12:44	URGENT
5	C1005	P1005	20258	Health	2025-03-26 23:14:37	NORMAL
6	C1006	P1006	47439	Accident	2025-02-16 09:43:36	URGENT
7	C1007	P1007	8822	Accident	2025-08-05 14:44:09	NORMAL
8	C1008	P1008	15435	Accident	2025-03-31 10:37:01	URGENT
9	C1009	P1009	52623	Accident	2025-01-01 07:50:05	NORMAL
10	C1010	P1010	91114	Health	2025-05-16 06:00:51	URGENT

Query1

```
SELECT
    c.ClaimID,
    c.PolicyNumber,
    c.ClaimType,
    c.Timestamp,
    CASE
        WHEN c.Timestamp::DATE BETWEEN p.StartDate AND p.EndDate
            THEN 'Within Coverage'
        ELSE 'Outside Coverage'
    END AS CoverageStatus
FROM claims c
JOIN policies p ON c.PolicyNumber = p.PolicyNumber
WHERE c.Timestamp::DATE NOT BETWEEN p.StartDate AND p.EndDate;
```

Data Output | Messages | Notifications

Showing rows: 1 to 34

	claimid [PK] text	policynumber text	claimtype text	timestamp timestamp without time zone	coveragestatus text
1	C1001	P1001	Fire	2025-07-28 07:13:47	Outside Coverage
2	C1003	P1003	Fire	2025-01-22 13:53:42	Outside Coverage
3	C1007	P1007	Accident	2025-08-05 14:44:09	Outside Coverage
4	C1009	P1009	Accident	2025-01-01 07:50:05	Outside Coverage
5	C1010	P1010	Health	2025-05-16 06:00:51	Outside Coverage
6	C1012	P1012	Health	2025-08-04 16:30:54	Outside Coverage
7	C1013	P1013	Accident	2025-07-11 01:16:11	Outside Coverage
8	C1014	P1014	Theft	2025-04-14 05:17:24	Outside Coverage
9	C1017	P1017	Health	2025-05-21 19:34:40	Outside Coverage

Query2

```
WITH customer_claims AS (
    SELECT
        cu.Region,
        cu.CustomerName,
        SUM(cl.ClaimAmount) AS TotalClaimAmount
    FROM claims cl
    JOIN policies po ON cl.PolicyNumber = po.PolicyNumber
    JOIN customers cu ON po.CustomerID = cu.CustomerID
    GROUP BY cu.Region, cu.CustomerName
),
ranked_claims AS (
    SELECT
        Region,
        CustomerName,
        TotalClaimAmount,
```

```
        DENSE_RANK() OVER (PARTITION BY Region ORDER BY TotalClaimAmount DESC)
AS RankInRegion
    FROM customer_claims
)
SELECT Region, CustomerName, TotalClaimAmount, RankInRegion
FROM ranked_claims
WHERE RankInRegion <= 2
ORDER BY Region, RankInRegion;
```

	region character varying (50)	customername character varying (100)	totalclaimamount bigint	rankinregion bigint
1	East	Customer_57	98349	1
2	East	Customer_63	97917	2
3	North	Customer_69	98314	1
4	North	Customer_55	96334	2
5	South	Customer_45	95338	1
6	South	Customer_50	94502	2
7	West	Customer_33	96345	1
8	West	Customer_85	93896	2

Query3

```
SELECT
    p.PolicyNumber,
    c.CustomerName,
    p.StartDate,
    p.EndDate
FROM policies p
JOIN customers c ON p.CustomerID = c.CustomerID
LEFT JOIN claims cl ON p.PolicyNumber = cl.PolicyNumber
WHERE CURRENT_DATE BETWEEN p.StartDate AND p.EndDate
  AND cl.ClaimID IS NULL;
```

Data Output Messages Notifications

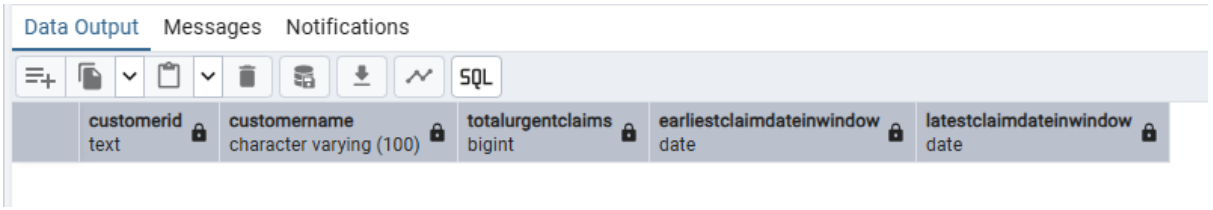
SQL

	policynumber text	customername character varying (100)	startdate date	enddate date

Query4

```
WITH urgent_claims AS (
    SELECT
        cl.ClaimID,
```

```
        cl.PolicyNumber,
        cl.Timestamp::DATE AS ClaimDate,
        po.CustomerID,
        cu.CustomerName
    FROM claims cl
    JOIN policies po ON cl.PolicyNumber = po.PolicyNumber
    JOIN customers cu ON po.CustomerID = cu.CustomerID
    WHERE cl.PriorityFlag = 'URGENT'
),
rolling_windows AS (
    SELECT
        u1.CustomerID,
        u1.CustomerName,
        COUNT(u2.ClaimID) AS TotalUrgentClaims,
        MIN(u2.ClaimDate) AS EarliestClaimDateInWindow,
        MAX(u2.ClaimDate) AS LatestClaimDateInWindow
    FROM urgent_claims u1
    JOIN urgent_claims u2
      ON u1.CustomerID = u2.CustomerID
     AND u2.ClaimDate BETWEEN u1.ClaimDate AND u1.ClaimDate + INTERVAL '30 days'
    GROUP BY u1.CustomerID, u1.CustomerName, u1.ClaimDate
)
SELECT
    CustomerID,
    CustomerName,
    TotalUrgentClaims,
    EarliestClaimDateInWindow,
    LatestClaimDateInWindow
FROM rolling_windows
WHERE TotalUrgentClaims > 2;
```



Query5

```
SELECT
    cl.ClaimID,
    cl.ClaimType,
    cl.ClaimAmount,
    po.PremiumAmount,
    ROUND(cl.ClaimAmount::NUMERIC / NULLIF(po.PremiumAmount, 0), 2) AS Ratio,
    RANK() OVER (
        PARTITION BY cl.ClaimType
        ORDER BY cl.ClaimAmount::NUMERIC / NULLIF(po.PremiumAmount, 0) DESC
```

```
    ) AS RankInType
FROM claims cl
JOIN policies po ON cl.PolicyNumber = po.PolicyNumber;
```

	claimid text	claimtype text	claimamount integer	premiumamount integer	ratio numeric	rankintype bigint
1	C1060	Accident	60235	7991	7.54	1
2	C1075	Accident	36149	5160	7.01	2
3	C1045	Accident	95338	15581	6.12	3
4	C1022	Accident	62056	16527	3.75	4
5	C1059	Accident	45647	14075	3.24	5
6	C1009	Accident	52623	20289	2.59	6
7	C1013	Accident	76018	38286	1.99	7
8	C1072	Accident	59925	33174	1.81	8
9	C1004	Accident	38606	21596	1.79	9

Total rows: 100 | Query complete 00:00:00.180

Query6

```
-- Step 1: Find customers whose AvgClaimAmount > 2 * AvgPremiumAmount
WITH customer_stats AS (
    SELECT
        cu.CustomerID,
        cu.CustomerName,
        AVG(cl.ClaimAmount) AS AvgClaimAmount,
        AVG(po.PremiumAmount) AS AvgPremiumAmount
    FROM claims cl
    JOIN policies po ON cl.PolicyNumber = po.PolicyNumber
    JOIN customers cu ON po.CustomerID = cu.CustomerID
    GROUP BY cu.CustomerID, cu.CustomerName
    HAVING AVG(cl.ClaimAmount) > 2 * AVG(po.PremiumAmount)
),

-- Step 2: Get most recent claim priority separately
recent_claims AS (
    SELECT
        cu.CustomerID,
        cu.CustomerName,
        cl.PriorityFlag,
        cl.Timestamp,
        ROW_NUMBER() OVER (PARTITION BY cu.CustomerID ORDER BY cl.Timestamp
DESC) AS rn
    FROM claims cl
    JOIN policies po ON cl.PolicyNumber = po.PolicyNumber
    JOIN customers cu ON po.CustomerID = cu.CustomerID
```

```
)

-- Step 3: Final filter (only customers whose most recent claim is NORMAL)
SELECT
    cs.CustomerID,
    cs.CustomerName,
    ROUND(cs.AvgClaimAmount, 2) AS AvgClaimAmount,
    ROUND(cs.AvgPremiumAmount, 2) AS AvgPremiumAmount,
    rc.PriorityFlag AS MostRecentClaimPriority
FROM customer_stats cs
JOIN recent_claims rc ON cs.CustomerID = rc.CustomerID
WHERE rc.rn = 1
  AND rc.PriorityFlag = 'NORMAL';
```

	customerid character varying (20)	customername character varying (100)	avgclaimamount numeric	avgpremiumamount numeric	mostrecentclaimpriority text
1	CU1003	Customer_3	70344.00	31762.00	NORMAL
2	CU1009	Customer_9	52623.00	20289.00	NORMAL
3	CU1015	Customer_15	66540.00	21279.00	NORMAL
4	CU1016	Customer_16	56543.00	13570.00	NORMAL
5	CU1037	Customer_37	38249.00	19027.00	NORMAL
6	CU1045	Customer_45	95338.00	15581.00	NORMAL
7	CU1049	Customer_49	82980.00	7347.00	NORMAL
8	CU1050	Customer_50	94502.00	20589.00	NORMAL
9	CU1055	Customer_55	96334.00	9239.00	NORMAL

Total rows: 15 Query complete 00:00:00.213

Query7

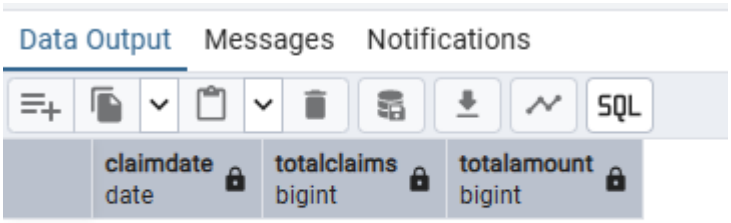
```
SELECT
    c1.Region AS RegionA,
    c2.Region AS RegionB,
    cl1.ClaimType,
    SUM(cl1.ClaimAmount + cl2.ClaimAmount) AS CombinedClaimAmount
FROM customers c1
JOIN policies p1 ON c1.CustomerID = p1.CustomerID
JOIN claims cl1 ON p1.PolicyNumber = cl1.PolicyNumber

JOIN customers c2 ON c2.CustomerID <> c1.CustomerID
JOIN policies p2 ON c2.CustomerID = p2.CustomerID
JOIN claims cl2 ON p2.PolicyNumber = cl2.PolicyNumber
                AND cl1.ClaimType = cl2.ClaimType

WHERE c1.Region < c2.Region   -- ensures RegionA ≠ RegionB and avoids duplicates
GROUP BY c1.Region, c2.Region, cl1.ClaimType
ORDER BY c1.Region, c2.Region, cl1.ClaimType;
```

	regiona character varying (50)	regionb character varying (50)	claimtype text	combinedclaimamount bigint
1	East	North	Accident	5646132
2	East	North	Fire	4051166
3	East	North	Health	1481634
4	East	North	Theft	3124819
5	East	South	Accident	4792663
6	East	South	Fire	3559103
7	East	South	Health	4561396
8	East	South	Theft	4812801
9	East	West	Accident	921599

Total rows: 24 Query complete 00:00:00.219

Query8

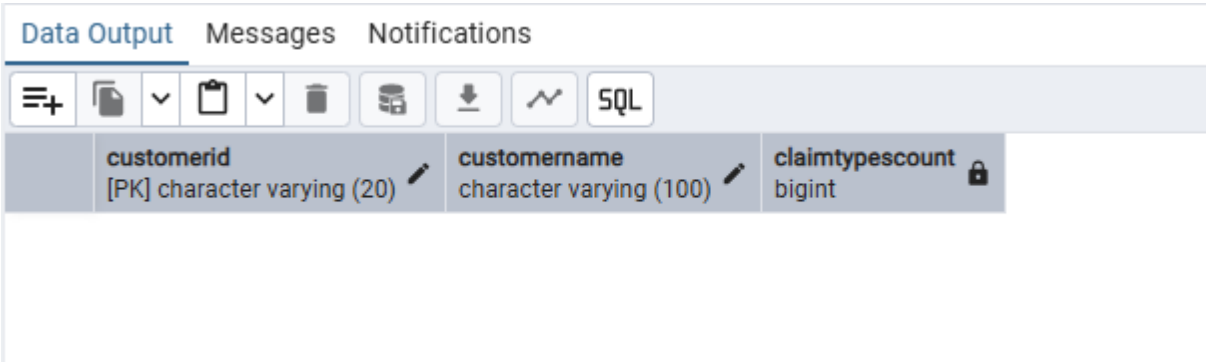
```
SELECT
    DATE(Timestamp) AS ClaimDate,
    COUNT(*) AS TotalClaims,
    SUM(ClaimAmount) AS TotalAmount
FROM claims
GROUP BY DATE(Timestamp)
HAVING COUNT(*) > 5
   AND SUM(ClaimAmount) > 500000
ORDER BY ClaimDate;
```



Query9

```
SELECT
    cu.CustomerID,
    cu.CustomerName,
    COUNT(DISTINCT cl.ClaimType) AS ClaimTypesCount
FROM customers cu
JOIN policies p ON cu.CustomerID = p.CustomerID
JOIN claims cl ON p.PolicyNumber = cl.PolicyNumber
GROUP BY cu.CustomerID, cu.CustomerName
HAVING
    COUNT(DISTINCT cl.ClaimType) >= 3   -- must cover at least 3 claim types
    AND NOT EXISTS (                    -- check they never claimed same type twice
```

```
    SELECT 1
    FROM claims cl2
    JOIN policies p2 ON cl2.PolicyNumber = p2.PolicyNumber
    WHERE p2.CustomerID = cu.CustomerID
    GROUP BY cl2.ClaimType
    HAVING COUNT(*) > 1            -- disqualify if any type claimed more than once
);
```



Query10

First Normal Form (1NF)

All three tables satisfy 1NF because:

- Each column holds **atomic (indivisible) values**.
- No repeating groups or arrays exist.

Examples:

- **Customers** → `CustomerID, CustomerName, Region, JoinDate` are all single values.
- **Policies** → `PolicyNumber, CustomerID, StartDate, EndDate, PremiumAmount, PolicyType` are atomic.
- **Claims** → `ClaimID, PolicyNumber, ClaimAmount, ClaimType, Timestamp, PriorityFlag` are atomic.

Second Normal Form (2NF)

Requirements: Table must be in 1NF **and** no partial dependency on a composite key.

- **Customers** → Primary Key = `CustomerID`. All attributes depend fully on it.
- **Policies** → Primary Key = `PolicyNumber`. All attributes (`CustomerID, StartDate, EndDate, PremiumAmount, PolicyType`) depend on it.
- **Claims** → Primary Key = `ClaimID`. All attributes (`PolicyNumber, ClaimAmount, ClaimType, Timestamp, PriorityFlag`) depend on it.

Since none of the tables have composite keys, partial dependency does not exist.

Third Normal Form (3NF)

Requirements: Table must be in 2NF **and** no transitive dependencies.

- **Customers** → `CustomerName, Region, JoinDate` depend only on `CustomerID`. Satisfies 3NF.
- **Claims** → `PolicyNumber, ClaimAmount, ClaimType, Timestamp, PriorityFlag` depend only on `ClaimID`. Satisfies 3NF.
- **Policies** → Possible issue:
    - If `PremiumAmount` is determined by `PolicyType` (e.g., "Health" always has a fixed premium), then `PremiumAmount` is **transitively dependent** on `PolicyType` instead of directly on `PolicyNumber`. May violate 3NF.