

# Sales Data Analysis

1. Write an SQL query to show the top 10 best-selling products by total Qty (i.e., quantity).

**ANSWER:**

```
SELECT PD."productkey", PD."ProductName", SUM("Qty") AS SumQty
FROM "Products" PD
JOIN "RetailSales" RS ON RS."ProductKey" = pd."productkey"
GROUP BY PD."productkey", PD."ProductName"
ORDER BY SumQty DESC
LIMIT 10;
```

Object Explorer

Publications

Schemas (1)

public

Aggregates

Collations

Domains

FTS Configurations

FTS Dictionaries

FTS Parsers

FTS Templates

Foreign Tables

Functions

Materialized Views

Operators

Procedures

Sequences

Tables (9)

Customers

Dates

Locations

Procurement

Products

Columns (4)

productkey

productName

Brands

Dept

Constraints

Indexes

RLS Policies

Rules

Triggers

Promotions

RetailSales

Columns (12)

Customerkey

productKey

StoreKey

DateKey

Timekey

Promokey

UnitPrice

Qty

postgres/postgres@PostgreSQL 16

Query

Query History

```
1 SELECT PD."productkey", PD."ProductName", SUM("Qty") AS SumQty
2 FROM "Products" PD
3 JOIN "RetailSales" RS ON RS."ProductKey" = pd."productkey"
4 GROUP BY PD."productkey", PD."ProductName"
5 ORDER BY SumQty DESC
6 LIMIT 10;
7
```

Data Output

Messages

Notifications

	productkey [PK] integer	ProductName character varying	sumqty bigint
1	98	Facemask M	6816
2	24	Candy Box 2PK	6797
3	62	All Org Quick Bar 10pc PK	6789
4	8	Egg 24 Extra LG	6788
5	116	65 IN TV Smart Nvue Brand	6788
6	7	Egg 12 Extra LG	6775
7	104	Conditioner Womens	6768
8	61	All Org Quick Bar 2pc PK	6748
9	110	Toothpast Clear	6748
10	100	Toilet Tissue Paper 15 Rolls	6738

Total rows: 10 of 10

Query complete 00:00:00.213

Ln 6, Col 10

2. Write an SQL query to show the total gross sales revenues each department makes by selling each brand.

**ANSWER:**

```
SELECT PD."Dept" AS DEPARTMENT, PD."Brands" AS BRANDS,SUM("GrossSales") AS
GrossSale
FROM "Products" PD
JOIN "RetailSales" RS ON RS."ProductKey" = pd."productkey"
GROUP BY PD."Dept", PD."Brands"
```

The screenshot shows a PostgreSQL IDE interface. On the left is the 'Object Explorer' showing the database schema. The main window displays a SQL query and its results in a table.

**Query:**

```
1 SELECT PD."Dept" AS DEPARTMENT, PD."Brands" AS BRANDS,SUM("GrossSales") AS GrossSale
2 FROM "Products" PD
3 JOIN "RetailSales" RS ON RS."ProductKey" = pd."productkey"
4 GROUP BY PD."Dept", PD."Brands"
5
```

**Data Output:**

	department character varying	brands character varying	grosssale real
1	Beauty&Health	Sbrand	269206
2	Electronics	TV2T	1.5330487e+07
3	Electronics	CREV	1.1846755e+07
4	Food	Fortunely	69538.4
5	Food	NWA	46392.62
6	Beauty&Health	Qlean	199138.72
7	Drinks	QualityEnergy	216741.17
8	Food	Marcus Brand	66782
9	Drinks	NWA	315783
10	Beauty&Health	AngelBrand	164088
11	Drinks	Marcus Brand	302100
12	Food	OFOOD	96259.85
13	Electronics	MP	3.022974e+06
14	Bakery	Kfood	52048.65
15	Food	GMH	102517.17
16	Egg&Diary	Sbrand	208018.3
17	Clothing	HF	78420
18	Bakery	OrcNatural	44125
19	Drinks	Kfood	137739
20	Food	Comel	12463.393
21	Clothing	CEM	314226
22	Drinks	Clear	126452.71
23	Food	AllOrig	118379.22
24	Food	FamilyHouse	132390.31
25	Drinks	BestVine	141425
26	Bakery	Sbrand	83772.77
27	Drinks	NYC Brand	24897.62
28	Electronics	Vshare	6.951839e+06

Total rows: 41 of 41    Query complete 00:00:00.086    Ln 4, Col 32

3. Write an SQL query to show the total net sales revenues each department makes in-store and online respectively.

**ANSWER:**

```
SELECT pd."Dept",
       SUM(CASE WHEN s."OLStore" = 'In-Store' THEN rs."NetSales" ELSE 0 END) AS
total_instore_revenue,
       SUM(CASE WHEN s."OLStore" = 'Online' THEN rs."NetSales" ELSE 0 END) AS
total_online_revenue
FROM "Products" PD
JOIN "RetailSales" RS ON RS."ProductKey" = pd."productkey"
JOIN "Stores" s ON RS."StoreKey" = s."StoreKey"
GROUP BY pd."Dept"
ORDER BY pd."Dept";
```

The screenshot shows a PostgreSQL IDE interface. On the left is the 'Object Explorer' pane showing a database schema with various objects like Aggregates, Domains, Functions, and Tables. The 'Tables' section is expanded, showing 'RetailSales' and its columns. The main pane displays a SQL query in the 'Query' tab. The query is the same one provided in the answer block. Below the query, the 'Data Output' pane shows the results of the query. The results are a table with three columns: 'Dept', 'total\_instore\_revenue', and 'total\_online\_revenue'. The data is sorted by department. The status bar at the bottom indicates 'Total rows: 7 of 7' and 'Query complete 00:00:00.118'.

Dept	total_instore_revenue	total_online_revenue
Bakery	174404.28	869.5164
Beauty&Health	1.02571156e+06	5394.6196
Clothing	1.1762212e+06	5077.6
Drinks	2.0227741e+06	9898.139
Egg&Diary	267547.16	2107.7205
Electronics	4.1309632e+07	238282.3
Food	1.00376275e+06	5720.3687

4. Write an SQL query to show how many customers are served by all cash registers of Store #1 per hour? (Note: StoreKey should be 1 because only Store #1's cash registers are of analytical interest. The fact table records data from each scan. So, you should not use SUM(Qty) because the question asks the total number of customers per hour, not the total number of scanned products per hour. Hint: The degenerate dimension can help!)

**ANSWER:**

```
SELECT T."Hour" AS Hour,
       COUNT(DISTINCT RS."CustomerKey") AS CustomersPerHour
FROM "Time" AS T
LEFT JOIN "RetailSales" AS RS
ON T."TimeKey" = RS."TimeKey"
WHERE RS."StoreKey" = 1
GROUP BY T."Hour";
```

The screenshot shows a PostgreSQL IDE interface. On the left, the Object Explorer displays the database schema, with the 'RetailSales' table selected under the 'public' schema. The main window shows the SQL query and its results.

**Query:**

```
1 SELECT T."Hour" AS Hour,
2      COUNT(DISTINCT RS."CustomerKey") AS CustomersPerHour
3 FROM "Time" AS T
4 LEFT JOIN "RetailSales" AS RS
5 ON T."TimeKey" = RS."TimeKey"
6 WHERE RS."StoreKey" = 1
7 GROUP BY T."Hour";
```

**Data Output:**

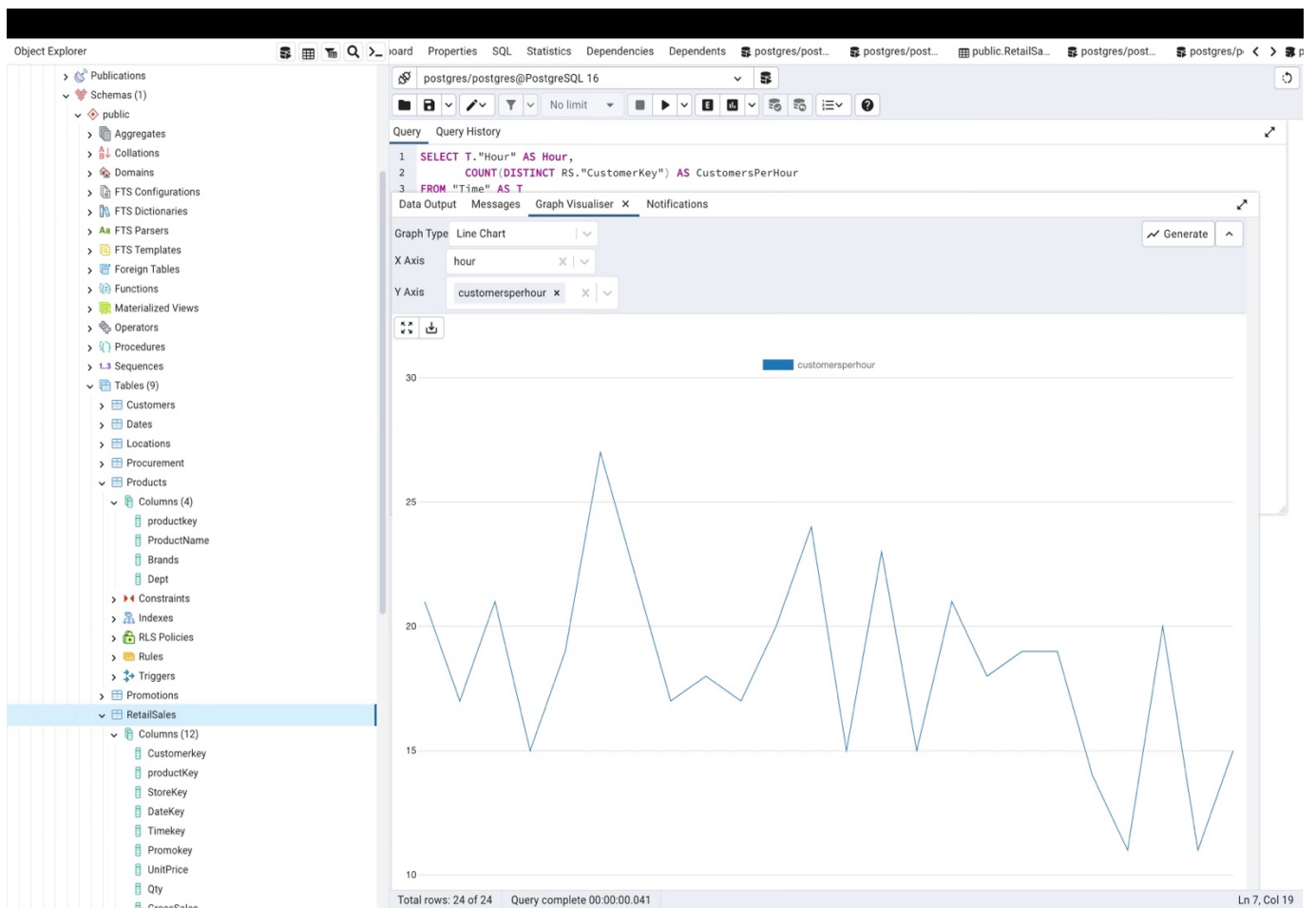
hour	customersperhour
1	0
2	1
3	2
4	3
5	4
6	5
7	6
8	7
9	8
10	9
11	10
12	11
13	12
14	13
15	14
16	15
17	16
18	17
19	18
20	19
21	20
22	21
23	22
24	23

Total rows: 24 of 24    Query complete 00:00:00.041    Ln 7, Col 19

5. Use “Graph Visualiser” from PostgreSQL to visualize the results of question 4. Make a screenshot of the chart and save it into your submission.

**ANSWER:**

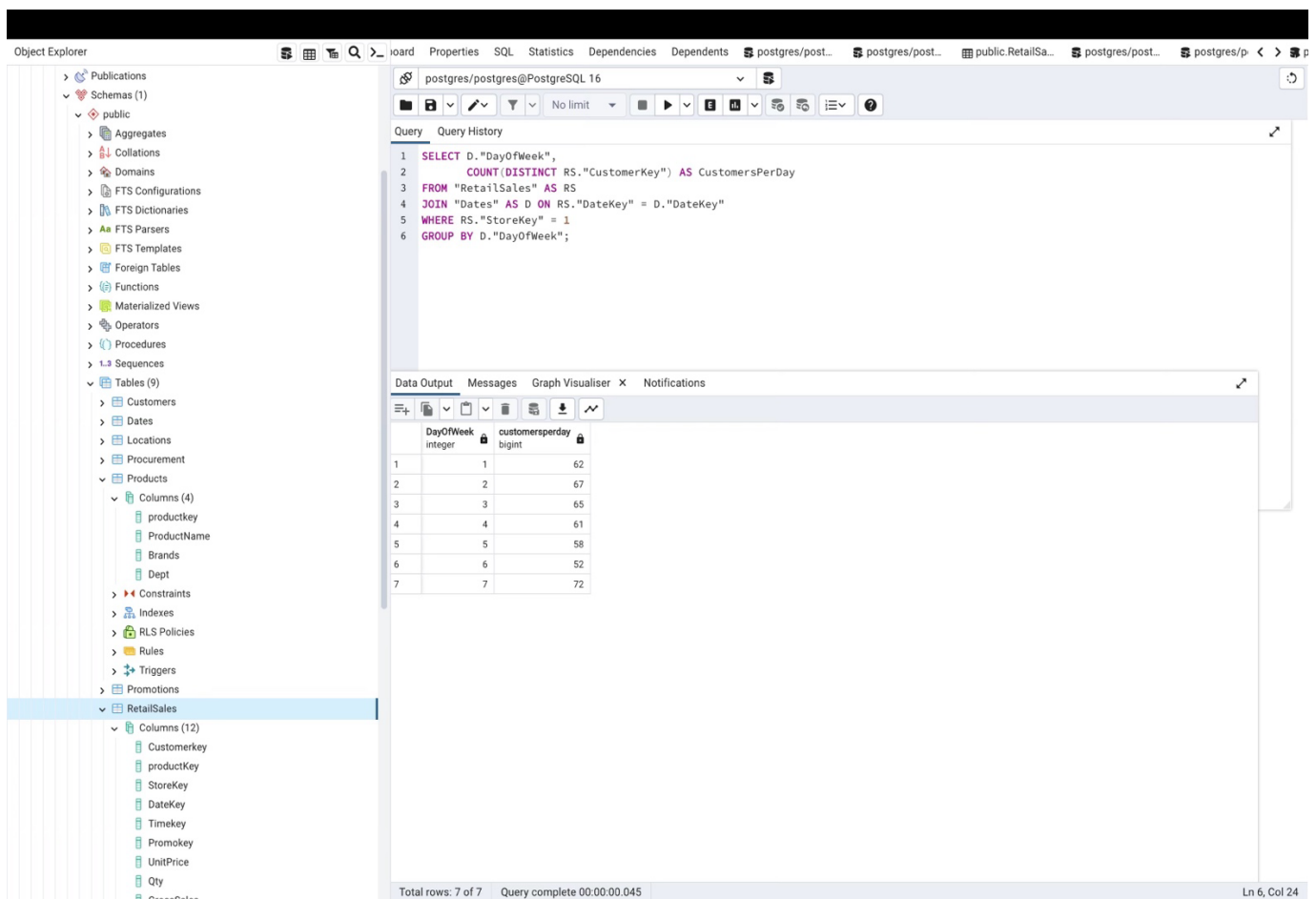
```
SELECT T."Hour" AS Hour,  
       COUNT(DISTINCT RS."CustomerKey") AS CustomersPerHour  
FROM "Time" AS T  
LEFT JOIN "RetailSales" AS RS  
ON T."TimeKey" = RS."TimeKey"  
WHERE RS."StoreKey" = 1  
GROUP BY T."Hour";
```



6. Write an SQL query to show how many customers are served by Store #1 in each day of a week. (Note: The SQL query should show the total number of customers served in Store #1 on Monday, Tuesday, ... Sunday. The number of customers and week day information must be in the same table. You should not write a query for Monday, and then another query for Tuesday, and then another query for Wednesday, etc. Using just one query to show the number of customers served by Store #1 in each day of a week is required).

**ANSWER:**

```
SELECT D."DayOfWeek",
       COUNT(DISTINCT RS."CustomerKey") AS CustomersPerDay
FROM "RetailSales" AS RS
JOIN "Dates" AS D ON RS."DateKey" = D."DateKey"
WHERE RS."StoreKey" = 1
GROUP BY D."DayOfWeek";
```



The screenshot shows a PostgreSQL IDE interface. On the left, the Object Explorer displays the database schema, with the 'RetailSales' table selected under the 'public' schema. The main query editor displays the following SQL query:

```
1 SELECT D."DayOfWeek",
2       COUNT(DISTINCT RS."CustomerKey") AS CustomersPerDay
3 FROM "RetailSales" AS RS
4 JOIN "Dates" AS D ON RS."DateKey" = D."DateKey"
5 WHERE RS."StoreKey" = 1
6 GROUP BY D."DayOfWeek";
```

Below the query editor, the Data Output pane shows the results of the query. The results are displayed in a table with two columns: 'DayOfWeek' (integer) and 'customersperday' (bigint). The data is as follows:

DayOfWeek	customersperday
1	62
2	67
3	65
4	61
5	58
6	52
7	72

The status bar at the bottom indicates 'Total rows: 7 of 7' and 'Query complete 00:00:00.045'.

**ANSWER:**

```
SELECT PD."ProductName", D."Quarter", SUM(rs."Qty") AS TotalQty
FROM "Products" AS PD
JOIN "RetailSales" AS RS ON PD."productkey" = rs."ProductKey"
JOIN "Dates" AS D ON RS."DateKey" = d."DateKey"
WHERE D."Quarter" = 4
AND PD."Dept" = 'Beauty&Health'
GROUP BY PD."ProductName", D."Quarter"
ORDER BY TotalQty DESC;
```

Object Explorer

Publications

Schemas (1)

public

Aggregates

Collations

Domains

FTS Configurations

FTS Dictionaries

FTS Parsers

FTS Templates

Foreign Tables

Functions

Materialized Views

Operators

Procedures

Sequences

Tables (9)

Customers

Dates

Locations

Procurement

Products

Columns (4)

productkey

productName

Brands

Dept

Constraints

Indexes

RLS Policies

Rules

Triggers

Promotions

RetailSales

Columns (12)

Customerkey

productKey

StoreKey

DateKey

Timekey

Promokey

UnitPrice

Qty

StoreSales

Properties

SQL

Statistics

Dependencies

Dependents

postgres/post...

postgres/post...

public.RetailSa...

postgres/post...

postgres/p...

postgres/postgres@PostgreSQL 16

Query

Query History

```

1 SELECT PD."productName", D."Quarter", SUM(rs."Qty") AS TotalQty
2 FROM "Products" AS PD
3 JOIN "RetailSales" AS RS ON PD."productkey" = rs."ProductKey"
4 JOIN "Dates" AS D ON RS."DateKey" = d."DateKey"
5 WHERE D."Quarter" = 4
6 AND PD."Dept" = 'Beauty&Health'
7 GROUP BY PD."productName", d."Quarter"
8 ORDER BY TotalQty DESC;

```

Data Output

Messages

Notifications

	productName character varying	Quarter integer	totalQty bigint
1	Toothpast Clear	4	1815
2	Facemask L	4	1727
3	Toothpast Super Mint	4	1698
4	Floss 32 pc	4	1688
5	Toothpast Mint	4	1684
6	Conditioner Womens	4	1682
7	Toilet Tissue Paper 6 Rolls	4	1676
8	Foundation Womens	4	1665
9	Qlean Brand Shampoo Womens	4	1587
10	Angel Brand Toilet Paper 6 Rolls	4	1550
11	Facemask M	4	1542
12	Toilet Tissue Paper 15 Rolls	4	1511
13	Shampoo Mens	4	1497
14	Angel Brand Toilet Paper 15 Rolls	4	1448
15	Towel White	4	1428
16	Conditioner Mens	4	1394

Total rows: 16 of 16

Query complete 00:00:00.041

Ln 8, Col 24

8. Write an SQL query to show whether there is a monthly increasing total GrossSales trend in the “Electronics” department.

**ANSWER:**

```
SELECT D."MonthOfYear",
       SUM(RS."GrossSales") AS GrossSales,
       LAG(SUM(RS."GrossSales")) OVER (ORDER BY D."MonthOfYear") AS
PreviousMonthGrossSales,
       CASE
         WHEN SUM(RS."GrossSales") > LAG(SUM(RS."GrossSales")) OVER (ORDER BY
D."MonthOfYear") THEN 'Increasing'
         WHEN SUM(RS."GrossSales") < LAG(SUM(RS."GrossSales")) OVER (ORDER BY
D."MonthOfYear") THEN 'Decreasing'
         ELSE 'No Change'
       END AS Trend
FROM "RetailSales" AS RS
JOIN "Products" AS P ON RS."ProductKey" = p."productkey"
JOIN "Dates" AS D ON RS."DateKey" = d."DateKey"
WHERE P."Dept" = 'Electronics'
GROUP BY D."MonthOfYear"
ORDER BY D."MonthOfYear";
```



Object Explorer

Publications

Schemas (1)

public

Aggregates

Collations

Domains

FTS Configurations

FTS Dictionaries

FTS Parsers

FTS Templates

Foreign Tables

Functions

Materialized Views

Operators

Procedures

Sequences

Tables (9)

Customers

Dates

Locations

Procurement

Products

Columns (4)

productkey

ProductName

Brands

Dept

Constraints

Indexes

RLS Policies

Rules

Triggers

Promotions

RetailSales

Columns (12)

Customerkey

productKey

StoreKey

DateKey

Timekey

Promokey

UnitPrice

Qty

RetailSales

postgres/postgres@PostgreSQL 16

Query Query History

```
1 SELECT D."MonthOfYear",
2       SUM(RS."GrossSales") AS GrossSales,
3       LAG(SUM(RS."GrossSales")) OVER (ORDER BY D."MonthOfYear") AS PreviousMonthGrossSales,
4       CASE
5         WHEN SUM(RS."GrossSales") > LAG(SUM(RS."GrossSales")) OVER (ORDER BY D."MonthOfYear") THEN 'Increasing'
6         WHEN SUM(RS."GrossSales") < LAG(SUM(RS."GrossSales")) OVER (ORDER BY D."MonthOfYear") THEN 'Decreasing'
7         ELSE 'No Change'
8       END AS Trend
9 FROM "RetailSales" AS RS
10 JOIN "Products" AS P ON RS."ProductKey" = P."productkey"
11 JOIN "Dates" AS D ON RS."DateKey" = D."DateKey"
12 WHERE P."Dept" = 'Electronics'
13 GROUP BY D."MonthOfYear"
14 ORDER BY D."MonthOfYear";
15
```

Data Output Messages Notifications

MonthOfYear	grosssales	previousmonthgrosssales	trend
integer	real	real	text
1	3.767726e+06	[null]	No Change
2	3.146072e+06	3.767726e+06	Decreasing
3	3.574447e+06	3.146072e+06	Increasing
4	3.377693e+06	3.574447e+06	Decreasing
5	3.679127e+06	3.377693e+06	Increasing
6	3.421717e+06	3.679127e+06	Decreasing
7	3.649379e+06	3.421717e+06	Increasing
8	3.649404e+06	3.649379e+06	Increasing
9	3.408127e+06	3.649404e+06	Decreasing
10	3.89456e+06	3.408127e+06	Increasing
11	3.465387e+06	3.89456e+06	Decreasing
12	3.536571e+06	3.465387e+06	Increasing

Total rows: 12 of 12    Query complete 00:00:00.057    Ln 14, Col 25