

# **QUERYING OF WALMART SALES DATABASE USING PostgreSQL**

# Importing .CSV data of walmart\_sales into PostgreSQL

The screenshot shows a Microsoft Excel spreadsheet titled "features" and a PostgreSQL query editor.

**Excel Spreadsheet:** The spreadsheet has columns labeled A through M. Column A is "Store", column B is "Date", and columns C through M contain various numerical and categorical data. A warning message at the top of the spreadsheet reads: "POSSIBLE DATA LOSS Some features might be lost if you save this workbook in the comma-delimited (.csv) format. To preserve these features, save it in an Excel (.xlsx) format."

**PostgreSQL Query Editor:** The connection is set to "walmart\_sales/postgres@PostgreSQL 15". The current query is:

```
1 CREATE TABLE features(
2   Store INTEGER,
3   Date TIMESTAMP,
4   Temperature float,
5   Fuel_Price VARCHAR(50),
6   Markdown1 VARCHAR(50),
7   Markdown2 VARCHAR(50),
8   Markdown3 VARCHAR(50),
9   Markdown4 VARCHAR(50),
10  Markdown5 VARCHAR(50),
11  CPI VARCHAR(50),
12  Unemployment VARCHAR(50),
13  IsHoliday boolean
14 )
```

The "Messages" tab shows the message: "Query returned successfully in 58 msec."

At the bottom, it says "Total rows: 0 of 0" and "Query complete 00:00:00.058".

# Importing .CSV data of walmart\_sales into PostgreSQL

The screenshot shows the pgAdmin 4 interface for managing a PostgreSQL 15 database named `walmart_sales`. The left sidebar displays the database structure, including servers, databases, and tables. The main area shows a query editor with the following SQL command:

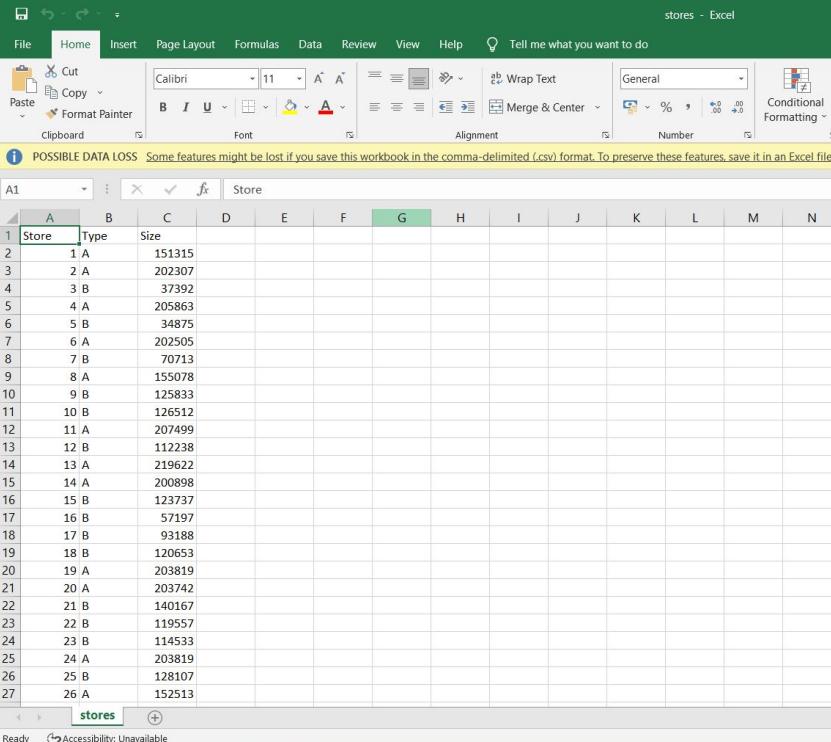
```
1 SELECT * FROM features
```

The Data Output tab below the query editor displays the results of the query, showing 7 rows of data from the `features` table. The columns and their data types are:

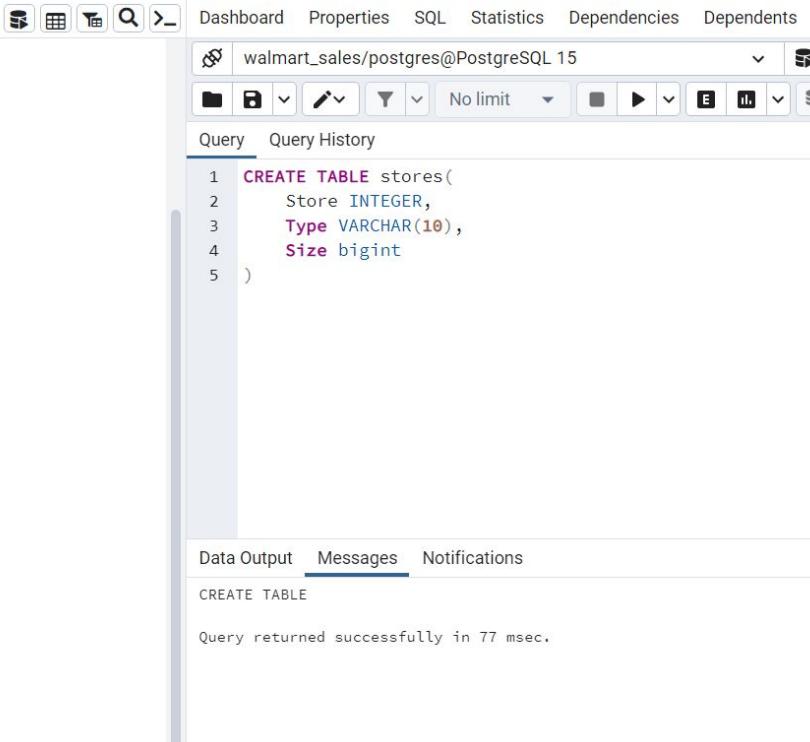
	store	date	temperature	fuel_price	markdown1	markdown2	markdown3
	integer	timestamp without time zone	double precision	character varying (50)	character varying (50)	character varying (50)	character varying (50)
1	1	2010-02-05 00:00:00	42.31	2.572	NA	NA	NA
2	1	2010-02-12 00:00:00	38.51	2.548	NA	NA	NA
3	1	2010-02-19 00:00:00	39.93	2.514	NA	NA	NA
4	1	2010-02-26 00:00:00	46.63	2.561	NA	NA	NA
5	1	2010-03-05 00:00:00	46.5	2.625	NA	NA	NA
6	1	2010-03-12 00:00:00	57.79	2.667	NA	NA	NA
7	1	2010-03-19 00:00:00	54.58	2.72	NA	NA	NA

The status bar at the bottom indicates "Total rows: 1000 of 8190 Query complete 00:00:00.251 Ln 1, Col 23".

# Importing .CSV data of walmart\_sales into PostgreSQL



A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	Store	Type	Size										
2	1	A	151315										
3	2	A	202307										
4	3	B	37392										
5	4	A	205863										
6	5	B	34875										
7	6	A	202505										
8	7	B	70713										
9	8	A	155078										
10	9	B	125833										
11	10	B	126512										
12	11	A	207499										
13	12	B	112238										
14	13	A	219622										
15	14	A	200898										
16	15	B	123737										
17	16	B	57197										
18	17	B	93188										
19	18	B	120653										
20	19	A	203819										
21	20	A	203742										
22	21	B	140167										
23	22	B	119557										
24	23	B	114533										
25	24	A	203819										
26	25	B	128107										
27	26	A	152513										



```
CREATE TABLE stores(
    Store INTEGER,
    Type VARCHAR(10),
    Size bigint
)
```

Data Output    Messages    Notifications

CREATE TABLE

Query returned successfully in 77 msec.

# Importing .CSV data of walmart\_sales into PostgreSQL

The screenshot shows the pgAdmin 4 interface for managing a PostgreSQL database named `walmart_sales/postgres@PostgreSQL 15*`. The left sidebar, titled "Browser", displays the database schema structure. Under the `walmart_sales` schema, the `stores` table is selected, showing its columns: `store` (integer), `type` (character varying (10)), and `size` (bigint). The table contains 45 rows, with data including store numbers 1 through 7, types A and B, and sizes ranging from 151315 to 70713. The main query editor window shows the command `SELECT* FROM stores`. The status bar at the bottom indicates "Total rows: 45 of 45" and "Query complete 00:00:00.090".

	store integer	type character varying (10)	size bigint
1	1	A	151315
2	2	A	202307
3	3	B	37392
4	4	A	205863
5	5	B	34875
6	6	A	202505
7	7	B	70713

# Importing .CSV data of walmart\_sales into PostgreSQL

The image shows a dual-monitor setup for importing data from a CSV file into a PostgreSQL database.

**Left Monitor (Excel):** A Microsoft Excel window titled "test - Excel" displays a CSV file named "walmart\_sales.csv". The data consists of 45 rows and 5 columns: Store, Dept, Date, IsHoliday, and empty columns E through K. The "IsHoliday" column contains boolean values (TRUE or FALSE). Row 1 is the header. A warning message at the top of the sheet says: "POSSIBLE DATA LOSS Some features might be lost if you save this workbook in the comma-delimited (.csv) format. To preserve these features, do not use the .csv file type when saving." The "Date" column uses the "YYYY-MM-DD" format.

**Right Monitor (pgAdmin 4):** A pgAdmin 4 interface window titled "pgAdmin 4" is connected to a PostgreSQL 15 database named "walmart\_sales/postgres@PostgreSQL 15".

- Browser:** Shows the database structure:
  - resume
  - walmart\_sales
    - Casts
    - Catalogs
    - Event Triggers
    - Extensions
    - Foreign Data Wrappers
    - Languages
    - Publications
  - Schemas (1)
    - public
      - Aggregates
      - Collations
      - Domains
      - FTS Configurations
      - FTS Dictionaries
      - FTS Parsers
      - FTS Templates
      - Foreign Tables
      - Functions
      - Materialized Views
      - Operators
      - Procedures
      - Sequences
    - Tables (2)
      - features
      - stores
        - Columns
        - Constraints
        - Indexes
        - RLS Policies

- Query Editor:** Displays the SQL command used to create the "test" table:

```
CREATE TABLE test(
    Store INTEGER,
    Dept INTEGER,
    Date TIMESTAMP,
    IsHoliday boolean
)
```
- Status:** Shows the query was executed successfully in 44 msec, with a total of 45 rows processed.

# Importing .CSV data of walmart\_sales into PostgreSQL

The screenshot shows the pgAdmin 4 interface for managing a PostgreSQL database named 'walmart\_sales' on 'PostgreSQL 15\*'. The left sidebar, titled 'Browser', displays the database schema structure, including catalogs, event triggers, extensions, foreign data wrappers, languages, publications, schemas (with 'public' expanded), aggregates, collations, domains, FTS configurations, FTS dictionaries, FTS parsers, FTS templates, foreign tables, functions, materialized views, operators, procedures, sequences, and tables (with 'test' selected). The main panel contains a query editor with the command 'SELECT \* FROM test' and a data output grid showing 7 rows of data from the 'test' table. The data includes columns: store (integer), dept (integer), date (timestamp without time zone), and isholiday (boolean). The bottom status bar indicates 'Total rows: 1000 of 115064' and 'Query complete 00:00:00.105'.

	store	dept	date	isholiday
1		1	2012-11-02 00:00:00	false
2		1	2012-11-09 00:00:00	false
3		1	2012-11-16 00:00:00	false
4		1	2012-11-23 00:00:00	true
5		1	2012-11-30 00:00:00	false
6		1	2012-12-07 00:00:00	false
7		1	2012-12-14 00:00:00	false

# Importing .CSV data of walmart\_sales into PostgreSQL

The screenshot illustrates the process of importing data from a Microsoft Excel spreadsheet into a PostgreSQL database using pgAdmin.

**Excel Spreadsheet (Left):** The Excel window shows a table named "train" with 27 rows of data. The columns are labeled A through H, and the data includes columns for Store, Dept, Date, Weekly\_Sales, and IsHoliday.

**pgAdmin Interface (Right):** The pgAdmin interface shows a connection to "walmart\_sales/postgres@PostgreSQL 15".

- Query Tab:** Displays the SQL command used to create the table:

```
CREATE TABLE train(
    Store INTEGER,
    Dept INTEGER,
    Date TIMESTAMP,
    Weekly_sales float,
    IsHoliday boolean
)
```
- Data Output Tab:** Shows the output of the CREATE TABLE command, indicating success: "Query returned successfully in 73 msec."
- Status Bar:** At the bottom, it displays "Total rows: 1000 of 115064" and "Query complete 00:00:00.073".

# Importing .CSV data of walmart\_sales into PostgreSQL

The screenshot shows the pgAdmin 4 interface for managing a PostgreSQL database named 'walmart\_sales'. The left sidebar, titled 'Browser', lists various database objects: Event Triggers, Extensions, Foreign Data Wrappers, Languages, Publications, Schemas (1), public (Aggregates, Collations, Domains, FTS Configurations, FTS Dictionaries, FTS Parsers, FTS Templates, Foreign Tables, Functions, Materialized Views, Operators, Procedures, Sequences), Tables (4) (features, stores, test, train), Trigger Functions, Types, Views, Subscriptions, Login/Group Roles, and Tablespaces. The 'train' table under 'Tables' is currently selected. The main workspace contains a query editor with the command 'SELECT \* FROM train' and a data output grid displaying 7 rows of sales data. The bottom status bar indicates 'Total rows: 1000 of 421570' and 'Query complete 00:00:00.593'.

pgAdmin 4

File Object Tools Help

Browser

- > Event Triggers
- > Extensions
- > Foreign Data Wrappers
- > Languages
- > Publications
- > Schemas (1)
  - > public
    - > Aggregates
    - > Collations
    - > Domains
    - > FTS Configurations
    - > FTS Dictionaries
    - > FTS Parsers
    - > FTS Templates
    - > Foreign Tables
    - > Functions
    - > Materialized Views
    - > Operators
    - > Procedures
    - > Sequences
  - > Tables (4)
    - > features
    - > stores
    - > test
    - > train
  - > Trigger Functions
  - > Types
  - > Views
  - > Subscriptions

walmart\_sales/postgres@PostgreSQL 15\*

Dashboard Properties SQL Statistics Dependencies Dependents Processes

Query History

```
1 SELECT * FROM train
2
```

Data Output Messages Notifications

	store	dept	date	weekly_sales	isholiday
	integer	integer	timestamp without time zone	double precision	boolean
1	1	1	2010-02-05 00:00:00	24924.5	false
2	1	1	2010-02-12 00:00:00	46039.49	true
3	1	1	2010-02-19 00:00:00	41595.55	false
4	1	1	2010-02-26 00:00:00	19403.54	false
5	1	1	2010-03-05 00:00:00	21827.9	false
6	1	1	2010-03-12 00:00:00	21043.39	false
7	1	1	2010-03-19 00:00:00	22136.64	false

Total rows: 1000 of 421570 Query complete 00:00:00.593

Ln 2, Col 1

The total number of data entries between "2010-02-05" and "2010-03-10" that were made on a business day for the shop and had a fuel price greater than 2.6.

The screenshot shows the pgAdmin 4 interface. The left sidebar is the 'Browser' pane, displaying database objects like Functions, Materialized Views, Operators, Procedures, Sequences, Tables (4), Constraints, Indexes, RLS Policies, Rules, Triggers, and stores. Under 'Tables (4)', 'features' is expanded, showing columns: store, date, temperature, fuel\_price, markdown1, markdown2, markdown3, markdown4, markdown5, cpi, unemployment, isholiday. Below 'features' are Constraints, Indexes, RLS Policies, Rules, Triggers, and 'stores'. 'stores' is expanded, showing columns: store, type, size. At the top, the title bar says 'pgAdmin 4' and 'File Object Tools Help'. The main area has tabs for 'Dashboard', 'Properties', 'SQL', 'Statistics', 'Dependencies', 'Dependents', 'Processes', and 'walmart\_sa...'. The 'SQL' tab is active, showing the following query:

```
1 SELECT COUNT(*)
2 FROM features
3 WHERE date >= '2010-02-05'
4 AND date <= '2010-03-10'
5 AND fuel_price >= '2.6'
6 AND isholiday = 'false'
7
```

Below the query is a 'Data Output' pane showing a single row of results:

count	bigint
1	521

At the bottom, it says 'Total rows: 1 of 1 Query complete 00:00:00.135' and 'Ln 4, Col 21'.

List of the top 5 largest stores according to their size and of 'C' type arranging in decreasing order.

The screenshot shows the pgAdmin 4 interface. The left sidebar displays a tree view of database objects under the 'Browser' tab, including 'store', 'date', 'temperature', 'fuel\_price', 'markdown1' through 'markdown5', 'cpi', 'unemployment', 'isholiday', and several system object categories like Constraints, Indexes, RLS Policies, Rules, and Triggers. Below this, the 'stores' table is expanded, showing its columns: 'store' (integer), 'type' (character varying(10)), and 'size' (bigint). The 'test' table is also listed with its own columns: 'store' and 'dept'. The main window shows a query editor with the following SQL code:

```
1 SELECT *
2 FROM stores
3 WHERE type = 'C'
4 ORDER BY size DESC
5 LIMIT 5
```

The results of the query are displayed in a table titled 'Data Output':

	store	type	size
	integer	character varying(10)	bigint
1	30	C	42988
2	43	C	41062
3	37	C	39910
4	44	C	39910
5	38	C	39690

At the bottom of the interface, status messages indicate "Total rows: 5 of 5" and "Query complete 00:00:00.082". The bottom right corner shows "Ln 4, Col 19".

# List of total sum of weekly sales of all stores date wise ranking from highest to lowest

The screenshot shows the pgAdmin 4 interface. On the left, the 'Browser' panel displays a tree view of database objects. Under the 'stores' schema, there are 'Columns (3)' with 'store', 'type', and 'size'. Under 'test', there are 'Columns (4)' with 'store', 'dept', 'date', and 'isholiday'. Under 'train', there are 'Columns (5)' with 'store', 'dept', 'date', 'weekly\_sales', and 'isholiday'. The 'Query' tab in the center contains a SQL query:

```
1 SELECT DATE(date), SUM(weekly_sales)
2 FROM train
3 GROUP BY DATE(date)
4 HAVING SUM(weekly_sales) >= 50000
5 ORDER BY SUM(weekly_sales)
```

The 'Data Output' tab at the bottom shows the results of the query:

	date	sum
1	2011-01-28	39599852.99000004
2	2012-01-27	39834974.66999994
3	2010-12-31	40432519
4	2011-01-21	40654648.02999999
5	2011-01-14	40673678.039999984
6	2010-09-24	41358514.410000026
7	2012-01-13	42023078.47999996

Total rows: 143 of 143    Query complete 00:00:00.149

Creating a list consisting of store, date, Consumer Price Index(CPI), type of store, department & size of the store where the size of store is less than 100000.

The screenshot shows the pgAdmin 4 interface with the following details:

- File Bar:** File, Object, Tools, Help
- Toolbar:** Browser, Dashboard, Properties, SQL, Statistics, Dependencies, Dependents, Processes
- Connection:** walmart\_sales/postgres@PostgreSQL 15\*
- Query Editor:** Contains the following SQL query:

```
1 SELECT features.store, features.date,cpi, type, size,dept
2 FROM features
3 INNER JOIN stores
4 ON features.store = stores.store
5 INNER JOIN test
6 ON features.store = test.store
7 WHERE size <= 100000
```
- Data Output:** Shows a table with 7 rows of data. The columns are: store, date, cpi, type, size, dept. The data is as follows:

	store	date	cpi	type	size	dept
1		3	214.4248812	B	37392	1
2		3	214.4248812	B	37392	1
3		3	214.4248812	B	37392	1
4		3	214.4248812	B	37392	1
5		3	214.4248812	B	37392	1
6		3	214.4248812	B	37392	1
7		3	214.4248812	B	37392	1

- Message Bar:** Total rows: 1000 of 5598502 Query complete 0:00:08.416 Ln 7, Col 21

# Creating a list consisting of fuel price of each day for type B stores of size greater than 50000 arranged from oldest to newest

The screenshot shows the pgAdmin 4 interface with the following details:

- File Object Tools Help** menu bar.
- Browser** pane on the left:
  - features** table:
    - Columns (12): store, date, temperature, fuel\_price, markdown1, markdown2, markdown3, markdown4, markdown5, cpi, unemployment, isholiday
    - Constraints
    - Indexes
    - RLS Policies
    - Rules
    - Triggers
  - stores** table:
    - Columns (3): store, type, size
    - Constraints
    - Indexes
    - RLS Policies
    - Rules
    - Triggers
  - test** table:
    - Columns
- Dashboard Properties SQL Statistics Dependencies Dependents Processes** tabs at the top.
- walmart\_sales/postgres@PostgreSQL 15\*** connection tab.
- Query History** tab.
- Query** tab containing the following SQL code:

```
1 SELECT fuel_price, AGE(date), type, size
2 FROM features
3 INNER JOIN stores
4 ON features.store = stores.store
5 WHERE type = 'B'
6 AND size >= '50000'
7 ORDER BY AGE(date) DESC
```
- Data Output** tab showing the results of the query:

	fuel_price character varying (50)	age interval	type character varying (10)	size bigint
1	2.572	13 years 1 mon 25 days	B	125833
2	2.58	13 years 1 mon 25 days	B	70713
3	2.788	13 years 1 mon 25 days	B	114533
4	2.666	13 years 1 mon 25 days	B	93188
5	2.788	13 years 1 mon 25 days	B	93638
6	2.962	13 years 1 mon 25 days	B	126512
7	2.788	13 years 1 mon 25 days	B	120653

- Messages Notifications** tabs.
- Total rows: 1000 of 2730 Query complete 00:00:00.087** status message.
- Successfully run. Total query runtime: 87 msec. 2730 rows affected.** success message.
- Scratch Pad** tab.

# Converting fuel\_price and weekly\_sales data types into double precision to make them useful for mathematical calculations from VARCHAR datatype .

The image shows two side-by-side DBeaver database client windows. Both windows have a top navigation bar with tabs for Dashboard, Properties, SQL, Statistics, Dependencies, Dependents, Processes, and a connection dropdown for 'walmart\_sales/postgres@PostgreSQL 15'. Below the navigation bar is a toolbar with various icons for file operations, search, and database management.

The left window displays a query in the SQL tab:

```
1 SELECT CAST(fuel_price AS double precision)
2 FROM features
```

The right window also displays a query in the SQL tab:

```
1 SELECT CAST(weekly_sales AS double precision)
2 FROM train
```

Both windows show a 'Data Output' tab at the bottom, which contains a table view of the results. The left window's table has a single column 'fuel\_price' with double precision type, containing 7 rows of data. The right window's table has a single column 'weekly\_sales' with double precision type, containing 7 rows of data. At the bottom of each window, there is a status bar indicating 'Total rows: 1000' and a timestamp like 'Query complete 00:00:00.049' or '0.264'. A green checkmark icon labeled 'Success' is visible in the bottom right corner of the right window's status bar.

# Creating a list which contains the dates when the weekly sales are greater than the overall average weekly sales for 72 & 92 departments arranged in the decreasing order of weekly sales.

The screenshot shows the pgAdmin 4 interface for a PostgreSQL database named 'walmart\_sales' on 'PostgreSQL 15\*'. The left sidebar displays the database schema with tables 'test' and 'train'. The 'test' table has columns: store, dept, date, and isholiday. The 'train' table has columns: store, dept, date, weekly\_sales, and isholiday. The main window shows a query editor with the following SQL code:

```
1 SELECT features.date, fuel_price, cpi, dept, weekly_sales
2 FROM features
3 INNER JOIN train
4 ON features.date = train.date
5 WHERE weekly_sales >= (SELECT AVG(weekly_sales)
6 FROM train)
7 AND dept IN (72,92)
8 ORDER BY weekly_sales DESC
9
10
11
```

The results of the query are displayed in a data output table:

	date	fuel_price	cpi	dept	weekly_sales
	timestamp without time zone	double precision	character varying (20)	integer	double precision
1	2010-11-26 00:00:00	3.039	182.7832769	72	693099.36
2	2010-11-26 00:00:00	2.735	211.7484333	72	693099.36
3	2010-11-26 00:00:00	3.186	132.8369333	72	693099.36
4	2010-11-26 00:00:00	3.039	204.9621	72	693099.36
5	2010-11-26 00:00:00	2.735	211.4062867	72	693099.36
6	2010-11-26 00:00:00	3.07	136.6895714	72	693099.36
7	2010-11-26 00:00:00	3.07	132.8369333	72	693099.36

Total rows: 1000 of 500760 | Query complete 00:00:00.728

Ln 7, Col 22

# Creating a table 'features\_new' to import .CSV data into the table which consists of data only after ' 2011-06-10' & when fuel price > 3 & it is a working day for the stores.

The screenshot shows the pgAdmin 4 interface for PostgreSQL 15. The left sidebar displays the database structure under the 'walmart\_sales' schema. The main pane shows a query editor with the following SQL code:

```
1 CREATE TABLE features_new(
2     store INTEGER,
3     date TIMESTAMP CHECK (date >= '2011-06-10'),
4     Temperarure double precision,
5     Fuel_price double precision CHECK (Fuel_price >= 3),
6     Markdown1 VARCHAR(20),
7     Markdown2 VARCHAR(20),
8     Markdown3 VARCHAR(20),
9     Markdown4 VARCHAR(20),
10    Markdown5 VARCHAR(20),
11    CPI VARCHAR(20),
12    unemployment VARCHAR(20),
13    Isholiday boolean CHECK ( Isholiday = 'FALSE')
14 )
```

The 'Messages' tab at the bottom indicates that the query was successfully executed.

# Creating a list which shows sales category as HIGH SALES (<25000) / MODERATE SALES (25000-50000) / LOW SALES(>50000) for each date .

The screenshot shows the pgAdmin 4 interface for PostgreSQL 15. The left sidebar displays the database structure under the 'Browser' tab, including 'features', 'stores' (with 'Columns', 'Constraints', 'Indexes', 'RLS Policies', 'Rules', 'Triggers'), 'test' (with 'Columns (4)' containing 'store', 'dept', 'date', 'isholiday'), 'train' (with 'Columns (5)' containing 'store', 'dept', 'date', 'weekly\_sales', 'isholiday'), and 'Dashboard', 'Properties', 'SQL', 'Statistics', 'Dependencies', 'Dependents', 'Processes' tabs at the top.

The main area shows a query editor with the following SQL code:

```
1 SELECT date,
2      CASE
3          WHEN (weekly_sales <= 25000) THEN 'LOW SALES'
4          WHEN (weekly_sales > 25000 AND weekly_sales <= 50000) THEN 'MODERATE SALES'
5          ELSE 'HIGH SALES'
6      END AS Sales_category
7 FROM train
```

The 'Data Output' tab displays the results of the query:

	date	sales_category
	timestamp without time zone	text
1	2010-02-05 00:00:00	LOW SALES
2	2010-02-12 00:00:00	MODERATE SALES
3	2010-02-19 00:00:00	MODERATE SALES
4	2010-02-26 00:00:00	LOW SALES
5	2010-03-05 00:00:00	LOW SALES
6	2010-03-12 00:00:00	LOW SALES
7	2010-03-19 00:00:00	LOW SALES

At the bottom, the status bar shows 'Total rows: 1000 of 421570' and 'Query complete 00:00:00.184'. The bottom right corner indicates 'Ln 1, Col 13'.

# Creating a list which shows total number of each type of stores A, B & C

The screenshot shows the pgAdmin 4 interface with a database connection to 'walmart\_sales/postgres@PostgreSQL 15\*'. The left sidebar displays the schema browser for the 'stores' table, showing columns 'store', 'type', and 'size', along with constraints, indexes, RLS policies, rules, and triggers. The main area contains a SQL query window with the following code:

```
1 SELECT
2 SUM(CASE type
3 WHEN 'A' THEN 1
4 ELSE 0
5 END) AS no_of_Atypes,
6 SUM(CASE type
7 WHEN 'B' THEN 1
8 ELSE 0
9 END) AS no_of_Btypes,
10 SUM(CASE type
11 WHEN 'C' THEN 1
12 ELSE 0
13 END) AS no_of_Ctypes
14 FROM stores
15
```

The results are displayed in a Data Output tab, showing a single row with three columns: 'no\_of\_atypes' (22), 'no\_of\_btypes' (17), and 'no\_of\_ctypes' (6).

	no_of_atypes	no_of_btypes	no_of_ctypes
1	22	17	6

Total rows: 1 of 1    Query complete 00:00:00.060    Ln 9, Col 23

# Inserting the available data of August month of 2013 into features table of store 1 and store2 .

The screenshot shows the pgAdmin 4 interface. On the left is the 'Browser' pane, which lists various database objects like Event Triggers, Extensions, Foreign Data Wrappers, Languages, Publications, Schemas (1), and Tables (4). Under 'Tables (4)', the 'features' table is selected, showing its 12 columns: store, date, temperarure, fuel\_price, markdown1, markdown2, markdown3, and markdown4. The main area is the 'Query' tab of the 'Dashboard' pane, containing the following SQL code:

```
1 INSERT INTO features
2   (store,date,temperarure,fuel_price,cpi,isholiday)
3 VALUES
4 ('1','2013-08-02','79.31','3.53','224.28521','FALSE'),
5 ('1','2013-08-09','78.61','3.49','224.3571','FALSE'),
6 ('1','2013-08-16','79.65','3.4013','224.52451','FALSE'),
7 ('1','2013-08-23','77.67','3.528','224.8255','FALSE'),
8 ('1','2013-08-30','79.87','3.83','224.6521','FALSE'),
9 ('2','2013-08-02','77.98','3.84','224.8745','FALSE'),
10 ('2','2013-08-09','77.85','3.72','224.35412','FALSE'),
11 ('2','2013-08-16','77.98','3.84','224.8745','FALSE'),
12 ('2','2013-08-23','76.99','3.45','224.4513','FALSE'),
13 ('2','2013-08-30','77.25','3.59','224.51665','FALSE')
```

Below the query, the 'Data Output' tab shows the message: 'INSERT 0 10'. The status bar at the bottom indicates 'Total rows: 0 of 0' and 'Query complete 00:00:00.054'. A green success message box in the bottom right corner says 'Query returned successfully in 54 msec.' with a checkmark icon.



**THANK  
YOU**

**MOHAMMED SAMEERUDDIN**