Walmart Sales — SQL Queries and Outputs

This document contains SQL queries executed Walmart_Sales.csv dataset. For each item: a short description, the SQL query run (for MySQL), and the resulting output table.

Row count

Count total rows in the table to confirm load succeeded.

Query:

```
SELECT COUNT(*) AS total_rows FROM walmart_sales;
```

Output:

total_rows
6435

Date range

Find minimum and maximum dates present in the data.

Query:

```
SELECT MIN(Date) AS min_date, MAX(Date) AS max_date FROM walmart_sales;
```

Output:

min_date	max_date
2010-02-05	2012-10-26

Distinct stores

Count distinct store IDs in the dataset.

Query:

```
SELECT COUNT(DISTINCT Store) AS num_stores FROM walmart_sales;
```

Output:

num_stores
45

Global aggregate stats

Basic global aggregates for weekly sales and other numerics.

Query:

```
SELECT COUNT(*) AS n, AVG(Weekly_Sales) AS avg_weekly_sales, MIN(Weekly_Sales)
AS min_weekly_sales, MAX(Weekly_Sales) AS max_weekly_sales, AVG(Temperature) AS
avg_temperature, AVG(Fuel_Price) AS avg_fuel_price, AVG(CPI) AS avg_cpi,
AVG(Unemployment) AS avg_unemployment FROM walmart_sales;
```

Output:

n	avg_weekly_sales	min_weekly_sales	max_weekly_sales	avg_temperature	avg_fuel_price	avg_cpi	avg_unemployment
6435	1046964.8775617732	209986.25000	3818686.45	60.663782	3.358607	171.578394	7.999151

Per-store summary (top 20 by avg sales)

Per-store summary: weeks recorded, avg, min, max weekly sales. Top 20 by average sales.

Query:

```
SELECT Store, COUNT(*) AS weeks, AVG(Weekly_Sales) AS avg_sales,
MIN(Weekly_Sales) AS min_sales, MAX(Weekly_Sales) AS max_sales FROM
walmart_sales GROUP BY Store ORDER BY avg_sales DESC LIMIT 20;
```

Output:

Store	weeks	avg_sales	min_sales	max_sales
20	143	2107676.8703496507	1761016.51	3766687.43
4	143	2094712.9606993007	1762539.3	3676388.98
14	143	2020978.400979021	1479514.66	3818686.45
13	143	2003620.306293707	1633663.12	3595903.2
2	143	1925751.3355244761	1650394.44	3436007.68
10	143	1899424.572657342	1627707.31	3749057.69
27	143	1775216.201958042	1263534.86	3078162.08
6	143	1564728.1862937063	1261253.18	2727575.18
1	143	1555264.3975524479	1316899.31	2387950.2

39	143	1450668.129160839	1158698.44	2554482.84
19	143	1444999.0356643356	1181204.53	2678206.42
31	143	1395901.4370629368	1198071.6	2068942.97
23	143	1389864.4604895099	1016756.1	2734277.1
24	143	1356755.393566433	1057290.41	2386015.75
11	143	1356383.1244755238	1100418.69	2306265.36
28	143	1323522.2418181808	1079669.11	2026026.39
41	143	1268125.4188111883	991941.730000	2263722.68
32	143	1166568.1549650352	955463.840000	1959526.96
18	143	1084718.421048951	540922.940000	2027507.15
22	143	1028501.0389510491	774262.280000	1962445.04

Monthly sales totals (all stores)

Monthly totals across all stores (year, month, total_monthly_sales).

Query:

```
SELECT STRFTIME('%Y', Date) AS year, STRFTIME('%m', Date) AS month,
SUM(Weekly_Sales) AS total_monthly_sales FROM walmart_sales GROUP BY year,
month ORDER BY year, month;
```

Output:

year	month	total_monthly_sales
2010	02	190332983.04
2010	03	181919802.5
2010	04	231412368.0500001
2010	05	186710934.34000003
2010	06	192246172.36000004
2010	07	232580125.9800001
2010	08	187640110.89000002

2010	09	177267896.36999997
2010	10	217161824.02
2010	11	202853370.14
2010	12	288760532.71999997
2011	01	163703966.83000007
2011	02	186331327.86999992
2011	03	179356448.29000002
2011	04	226526510.9700001
2011	05	181648158.16
2011	06	189773385.19000003
2011	07	229911398.87000012
2011	08	188599332.24999988
2011	09	220847738.41999993
2011	10	183261283.15000007
2011	11	210162354.87
2011	12	288078102.4800002
2012	01	168894471.66000006
2012	02	192063579.54000008
2012	03	231509650.4899999
2012	04	188920905.95000002
2012	05	188766479.4499998
2012	06	240610329.28999984
2012	07	187509452.40000004
2012	08	236850765.67999992
2012	09	180645544.46999988
2012	10	184361680.41999996

Per-store holiday uplift (top 20)

Per-store difference between avg holiday-week sales and avg non-holiday-week sales. (Top 20 uplift)

Query:

```
SELECT h.Store,
            h.avg_holiday_sales,
            n.avg_nonholiday_sales,
            (h.avg_holiday_sales - n.avg_nonholiday_sales) AS holiday_diff
     FROM
        (SELECT Store, AVG(Weekly_Sales) AS avg_holiday_sales FROM
walmart_sales WHERE Holiday_Flag=1 GROUP BY Store) h
     JOIN
        (SELECT Store, AVG(Weekly_Sales) AS avg_nonholiday_sales FROM
walmart_sales WHERE Holiday_Flag=0 GROUP BY Store) n
     ON h.Store = n.Store
     ORDER BY holiday_diff DESC
     LIMIT 20;
```

Output:

Store	avg_holiday_sales	avg_nonholiday_sales	holiday_diff
10	2113755.949	1883309.431578947	230446.517421
28	1478244.605	1311888.9814285708	166355.623571
35	1074348.457	908099.154211	166249.302789
2	2079266.9	1914208.8118796984	165058.088120
4	2243102.624	2083555.8431578942	159546.780842
20	2249035.081	2097048.433458646	151986.647541
19	1577046.734	1435070.6372932333	141976.096707
12	1138140.42	999291.924436	138848.495564
24	1475098.2510000002	1347857.4343609018	127240.816639
27	1892299.2780000002	1766412.9631578953	125886.314842
6	1680907.927	1555992.8674436095	124915.059556
1	1665747.656	1546957.3856390982	118790.270361
13	2113043.806	1995392.9754887223	117650.830511

31	1500026.03	1388072.5203007518	111953.509699
7	672400.265000	562964.454812	109435.810188
39	1551127.48	1443114.7945112777	108012.685489
14	2120582.9979999997	2013489.333533835	107093.664466
11	1448394.485	1349464.9770676687	98929.507932
17	979796.971000	887099.015940	92697.955060
18	1169422.1609999998	1078349.7187969924	91072.442203

Train / Test split counts (example boundary 2012-01-01)

Create train/test split by date and show counts.

Query:

```
SELECT SUM(CASE WHEN Date < '2012-01-01' THEN 1 ELSE 0 END) AS train_rows,
SUM(CASE WHEN Date >= '2012-01-01' THEN 1 ELSE 0 END) AS test_rows FROM
walmart_sales;
```

Output:

train_rows	test_rows
4500	1935

Yearly sales per store (sample top 50 pct change consecutive years)

Compute yearly sales per store and pct change year-over-year between consecutive years. Show top 50 increases.

Query:

```
WITH yearly AS (
       SELECT Store, STRFTIME('%Y', Date) AS yr, SUM(Weekly_Sales) AS
yearly_sales
       FROM walmart_sales
       GROUP BY Store, yr
     )
     SELECT y1.Store, y1.yr AS year, y1.yearly_sales, y2.yearly_sales AS
next_year_sales,
            (y2.yearly_sales - y1.yearly_sales) / NULLIF(y1.yearly_sales,0) AS
pct_change
     FROM yearly y1
```

```
    JOIN yearly y2 ON y1.Store = y2.Store AND CAST(y2.yr AS INTEGER) =
CAST(y1.yr AS INTEGER) + 1
    ORDER BY pct_change DESC
    LIMIT 50;
```

Output:

Store	year	yearly_sales	next_year_sales	pct_change
38	2010	16587794.520000001	19940758.45	0.202134
7	2010	25568078.149999995	30662640.52	0.199255
4	2010	95680470.81000003	111092293.32999998	0.161076
41	2010	57738220.65	66715874.31	0.155489
39	2010	65782276.319999985	75777603.30000003	0.151946
42	2010	25498089.909999996	29117302.670000006	0.141941
9	2010	25129219.760000005	28685969.650000002	0.141538
44	2010	13607519.449999997	15498194.669999996	0.138943
31	2010	65560272.75999999	74169225.52000001	0.131314
17	2010	41104920.440000005	46391839.760000005	0.128620
32	2010	55190936.68000001	61347193.47999999	0.111545
3	2010	18745419.000000007	20816876.57	0.110505
5	2010	14836030.769999998	16470820.000000002	0.110190
16	2010	24728632.590000004	27421367.48999999	0.108891
37	2010	24508469.859999996	27081495.770000007	0.104985
1	2010	73278832.00000003	80921918.83	0.104301
8	2010	43204474.84	47512786.16	0.099719
13	2010	95272735.44999999	104537513.32999997	0.097245
34	2010	46150416.79	50360182.05999998	0.091218
12	2010	48370383.860000014	52582000.57000001	0.087070
40	2010	46357359.519999996	50340542.92999999	0.085923

11	2010	65255138.23000001	70523582.88999999	0.080736
20	2010	101733080.72000001	109837002.36000001	0.079659
26	2010	48390697.78	52049251.72	0.075604
21	2010	37631108.269999996	40234883.940000005	0.069192
28	2010	64778764.93000001	69156008.58	0.067572
45	2010	38536343.370000005	41135367.88	0.067443
23	2010	67709105.30999997	72273533.79	0.067412
22	2010	50865280.410000004	53554711.92999999	0.052874
10	2010	94472202.21	98916894.74000002	0.047048
6	2010	76912320.69000001	80528762.94999999	0.047020
24	2010	66890648.24000002	69938976.87	0.045572
29	2010	26946827.490000002	27950345.200000003	0.037241
25	2010	35136267.73999999	36434405.72	0.036946
2	2010	95277864.18999998	98607881.41999999	0.034951
19	2010	72580528.63999999	74841900.18	0.031157
27	2010	90013176.57000001	91922684.19	0.021214
30	2010	21739086.630000003	22182148.409999996	0.020381
43	2010	31537005.989999995	32053195.44999999	0.016368
33	2010	12766834.26	12957836.67	0.014961
15	2010	32023528.309999987	32282624.900000013	0.008091
14	2010	105462242.38	106096270.70000003	0.006012
18	2010	55978417.3	54217740.11	-0.031453
38	2011	19940758.45	18631073.450000003	-0.065679
44	2011	15498194.669999996	14187373.72	-0.084579
36	2010	21153125.69	18972618.75	-0.103082
33	2011	12957836.67	11435551.03	-0.117480

39	2011	75777603.30000003	65885662.84999999	-0.130539
17	2011	46391839.760000005	40285378.63	-0.131628
3	2011	20816876.57	18024439.499999996	-0.134143