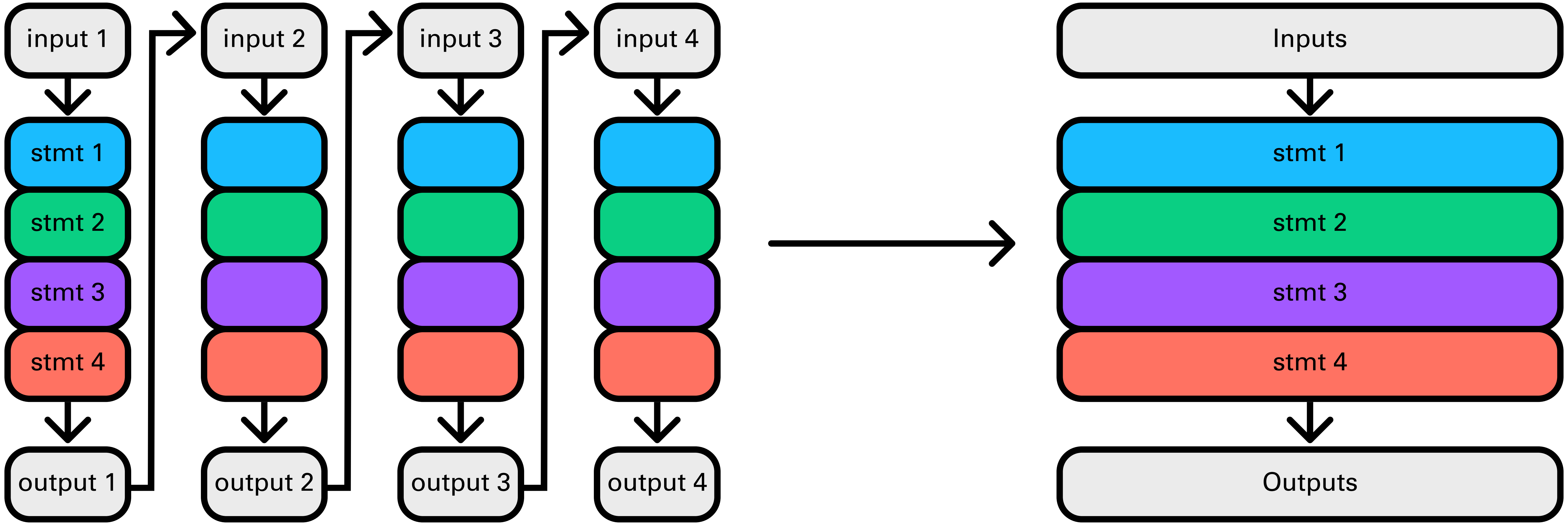
Efficiently Executing UDFs via Batching

Kai Franz

Batching UDFs

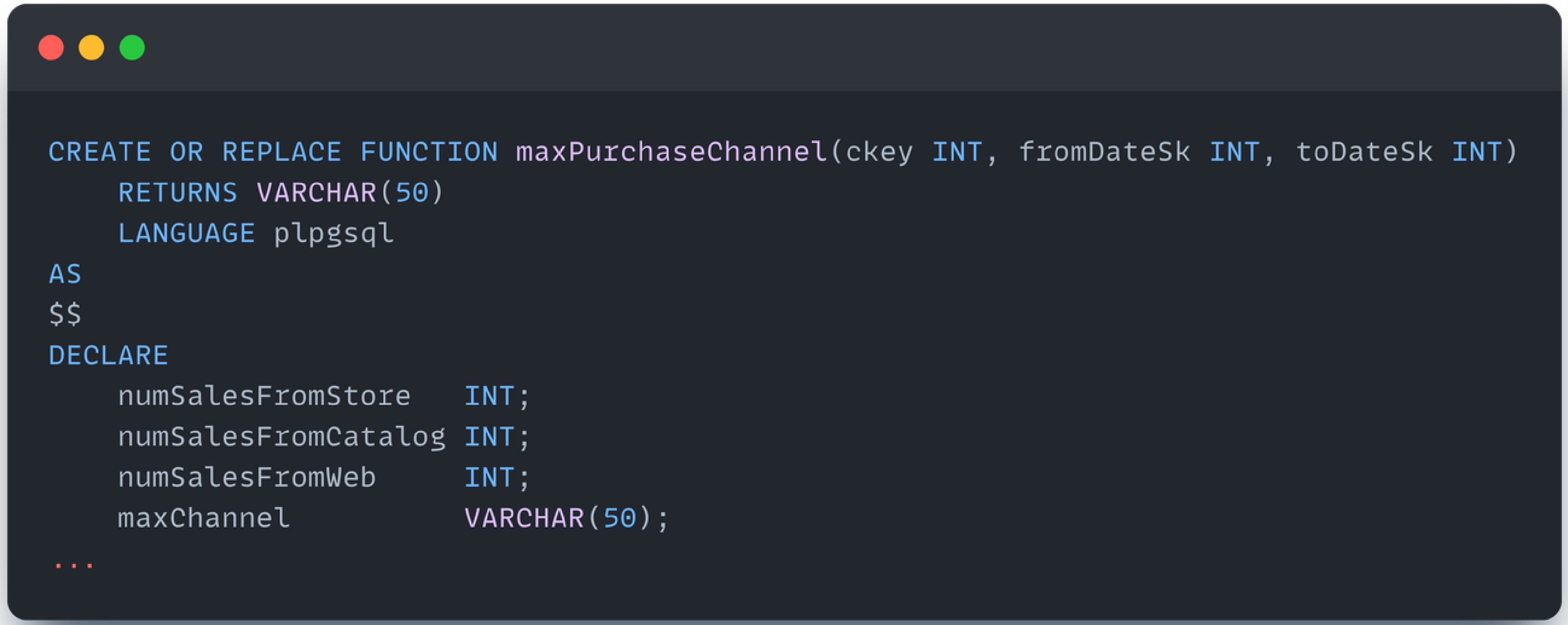
- Naive UDF execution: process one row at a time

- Batching: apply each statement to all rows at the same time
- DBMS is good at set-oriented processing

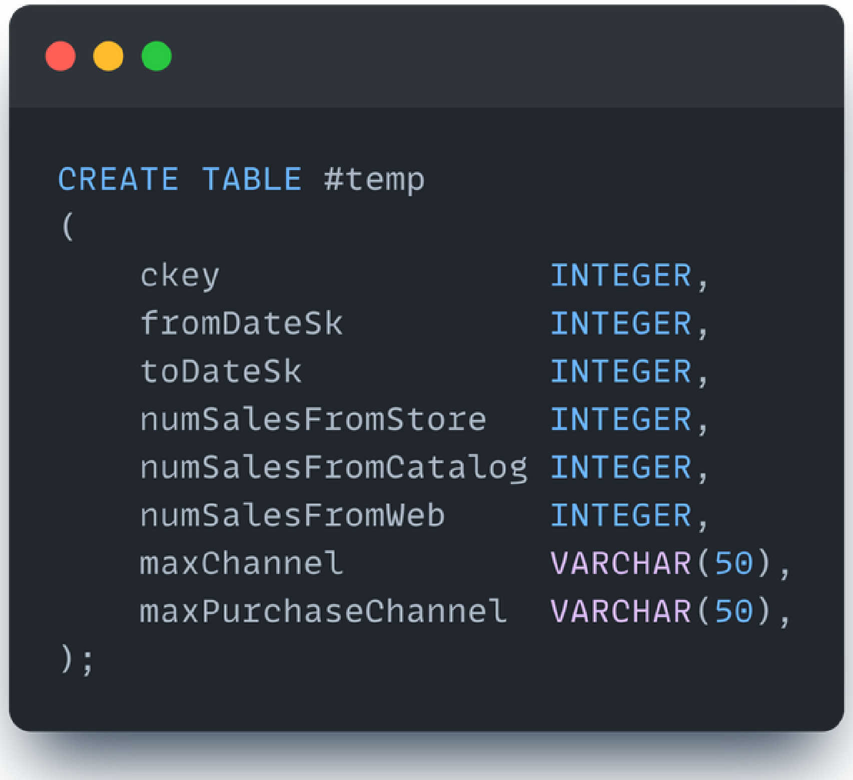


Parameters & Local Variables

- Create temp table with a column for each variable



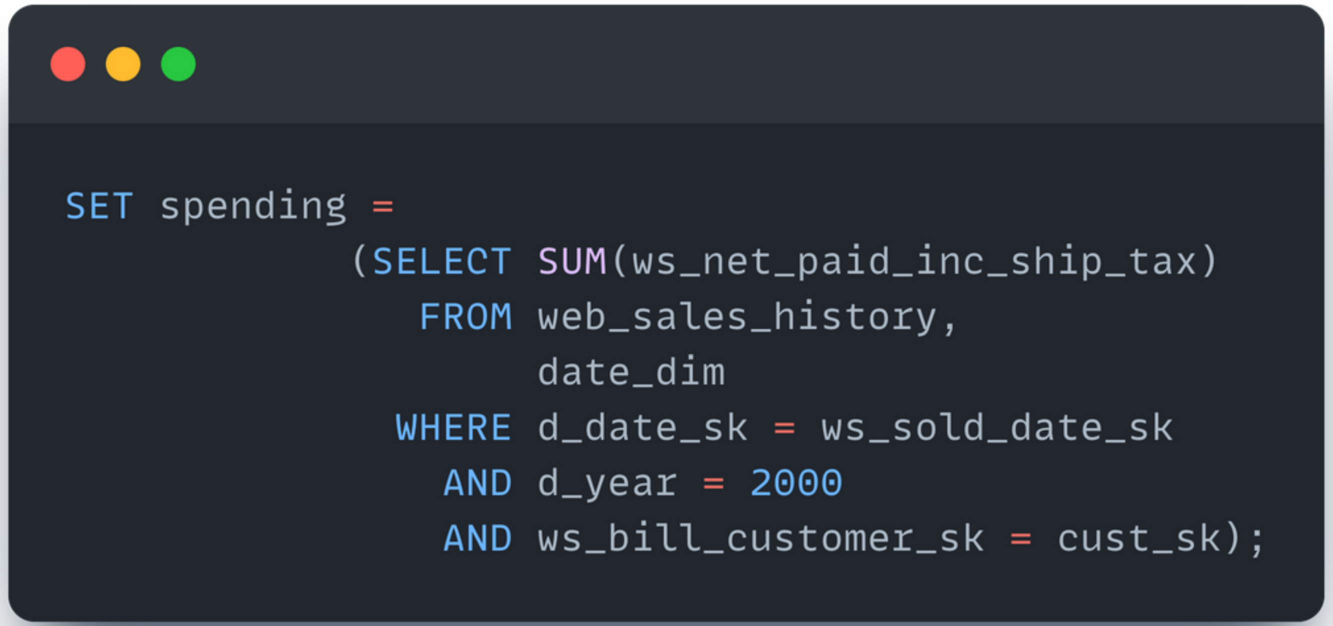
```
CREATE OR REPLACE FUNCTION maxPurchaseChannel(ckey INT, fromDateSk INT, toDateSk INT)
    RETURNS VARCHAR(50)
    LANGUAGE plpgsql
AS
$$
DECLARE
    numSalesFromStore   INT;
    numSalesFromCatalog INT;
    numSalesFromWeb     INT;
    maxChannel          VARCHAR(50);
...
```



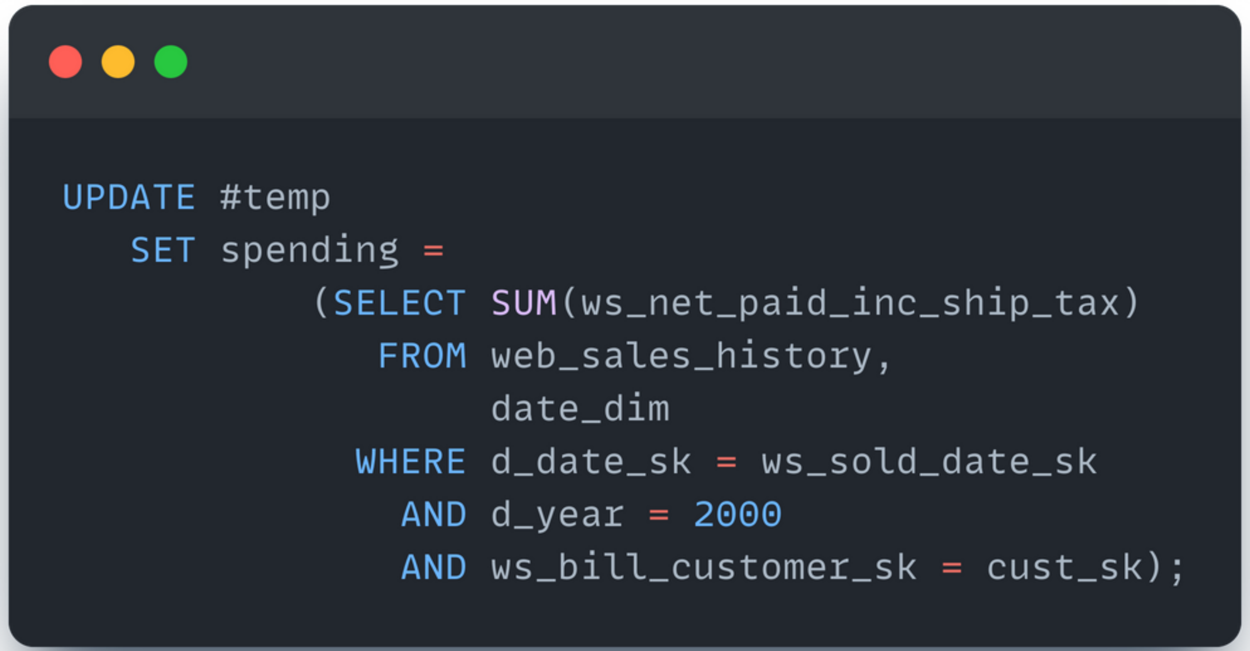
```
CREATE TABLE #temp
(
    ckey                INTEGER,
    fromDateSk          INTEGER,
    toDateSk            INTEGER,
    numSalesFromStore   INTEGER,
    numSalesFromCatalog INTEGER,
    numSalesFromWeb     INTEGER,
    maxChannel          VARCHAR(50),
    maxPurchaseChannel  VARCHAR(50),
);
```

Writing to Local Variables

- SET, SELECT INTO, etc. become UPDATE



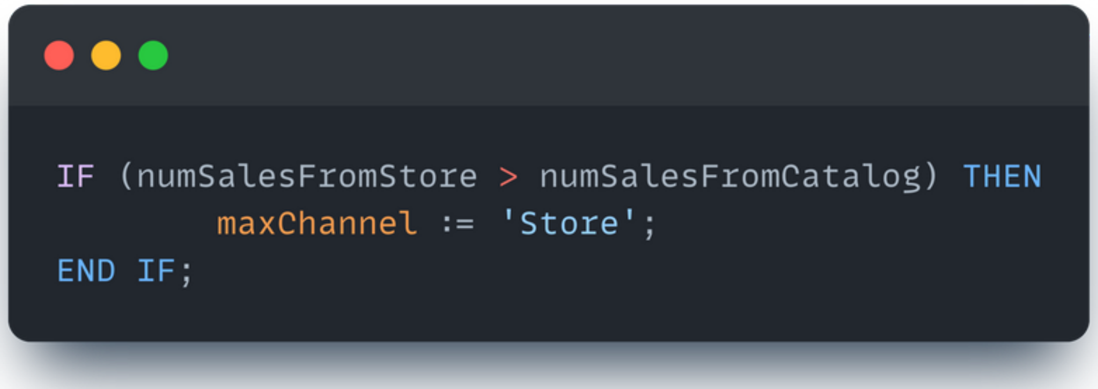
```
SET spending =
            (SELECT SUM(ws_net_paid_inc_ship_tax)
               FROM web_sales_history,
                    date_dim
              WHERE d_date_sk = ws_sold_date_sk
                AND d_year = 2000
                AND ws_bill_customer_sk = cust_sk);
```



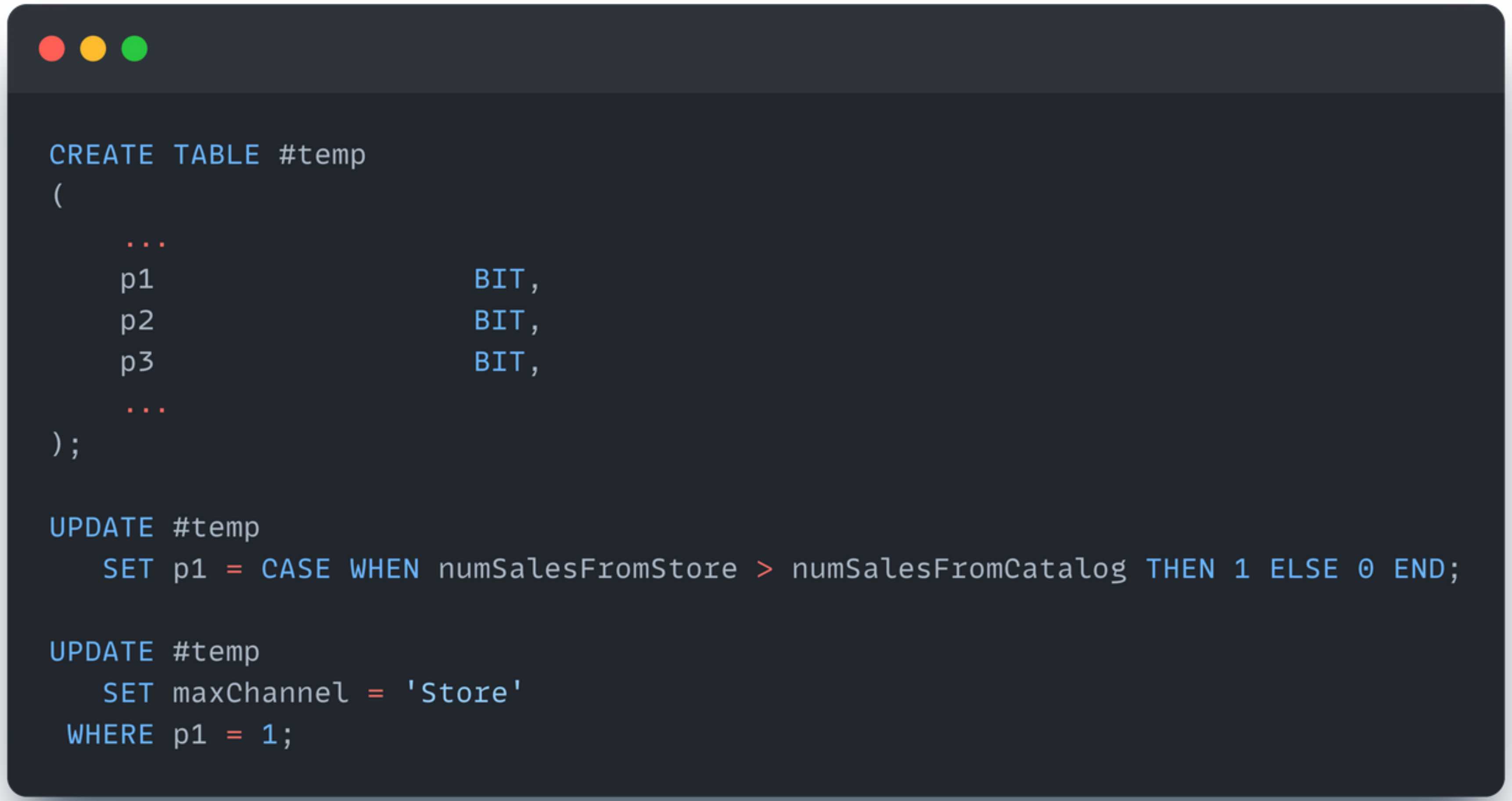
```
UPDATE #temp
   SET spending =
           (SELECT SUM(ws_net_paid_inc_ship_tax)
              FROM web_sales_history,
                   date_dim
             WHERE d_date_sk = ws_sold_date_sk
               AND d_year = 2000
               AND ws_bill_customer_sk = cust_sk);
```

Branching

- Froid-style boolean column for each branch predicate
- Statements inside branch guarded with WHERE



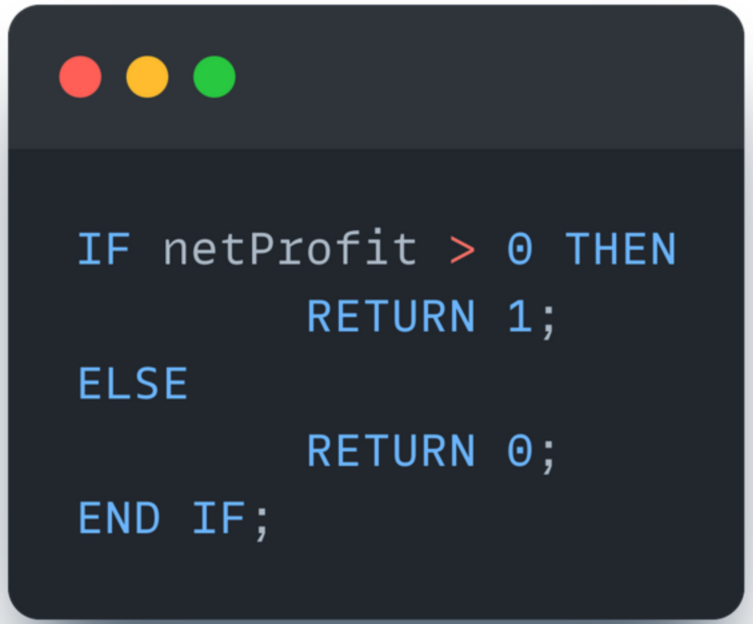
```
IF (numSalesFromStore > numSalesFromCatalog) THEN
        maxChannel := 'Store';
END IF;
```



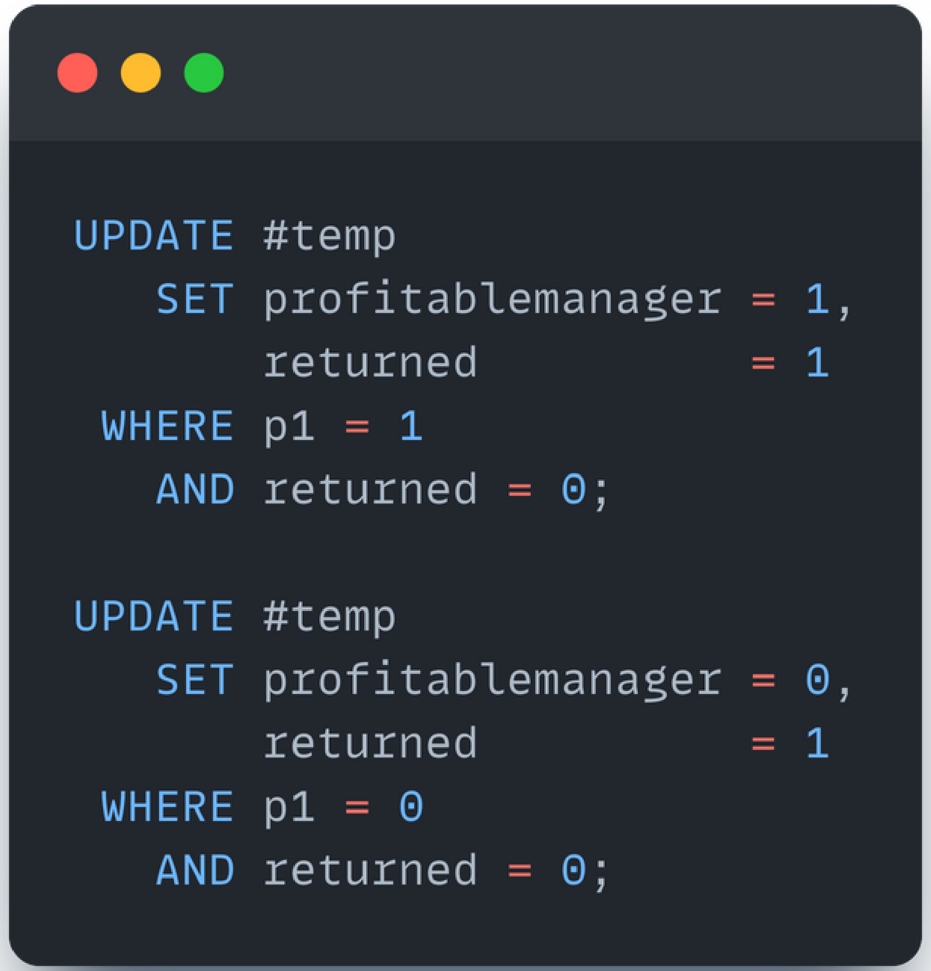
```
CREATE TABLE #temp
(
    ...
    p1                  BIT,
    p2                  BIT,
    p3                  BIT,
    ...
);

UPDATE #temp
   SET p1 = CASE WHEN numSalesFromStore > numSalesFromCatalog THEN 1 ELSE 0 END;

UPDATE #temp
   SET maxChannel = 'Store'
 WHERE p1 = 1;
```

Return Statements

- Boolean column tracks if row has returned yet



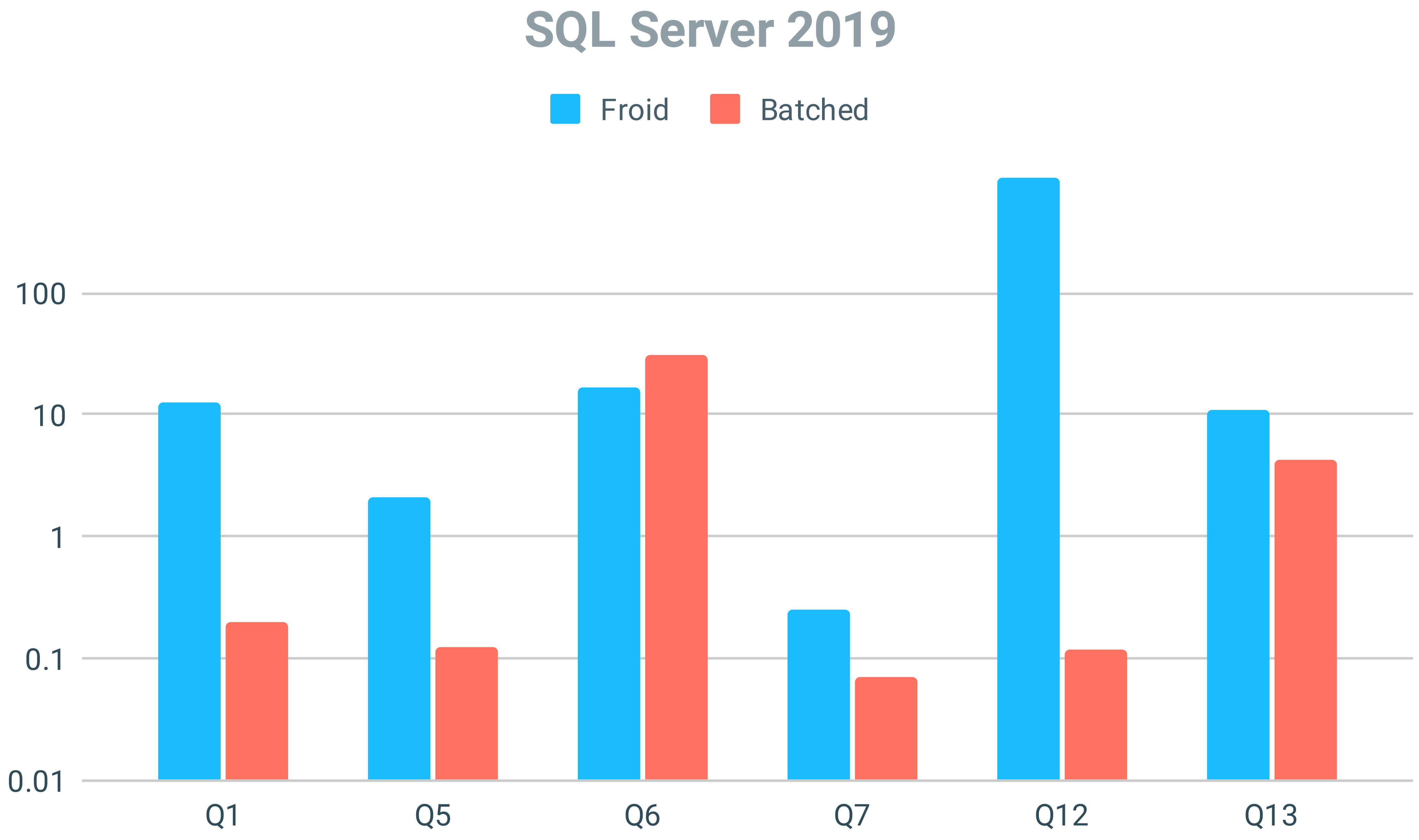
```
IF netProfit > 0 THEN
        RETURN 1;
ELSE
        RETURN 0;
END IF;
```



```
UPDATE #temp
   SET profitablemanager = 1,
       returned          = 1
 WHERE p1 = 1
   AND returned = 0;

UPDATE #temp
   SET profitablemanager = 0,
       returned          = 1
 WHERE p1 = 0
   AND returned = 0;
```

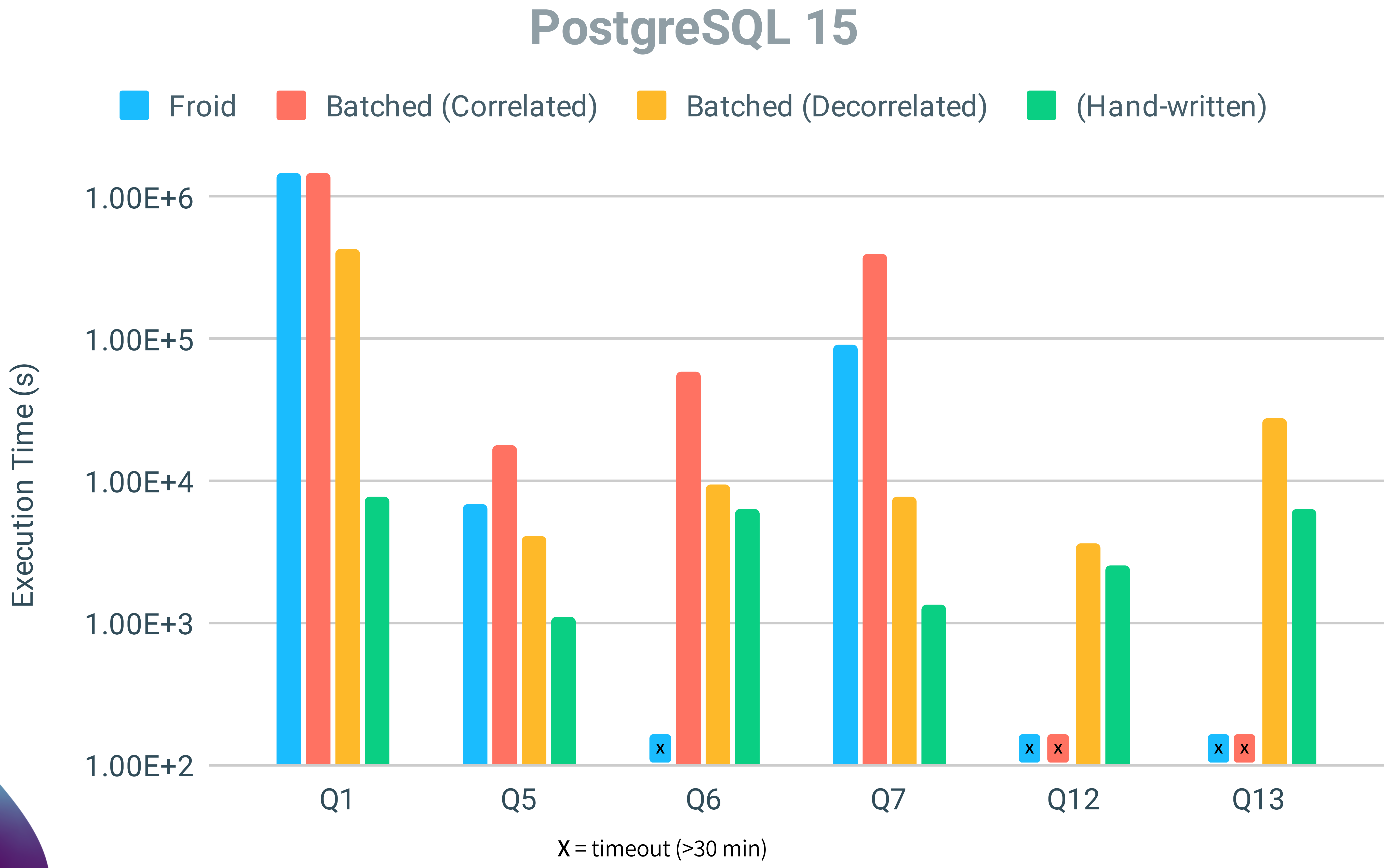
Experiments

- ProcBench benchmark, scale factor 1
- Compared against Froid



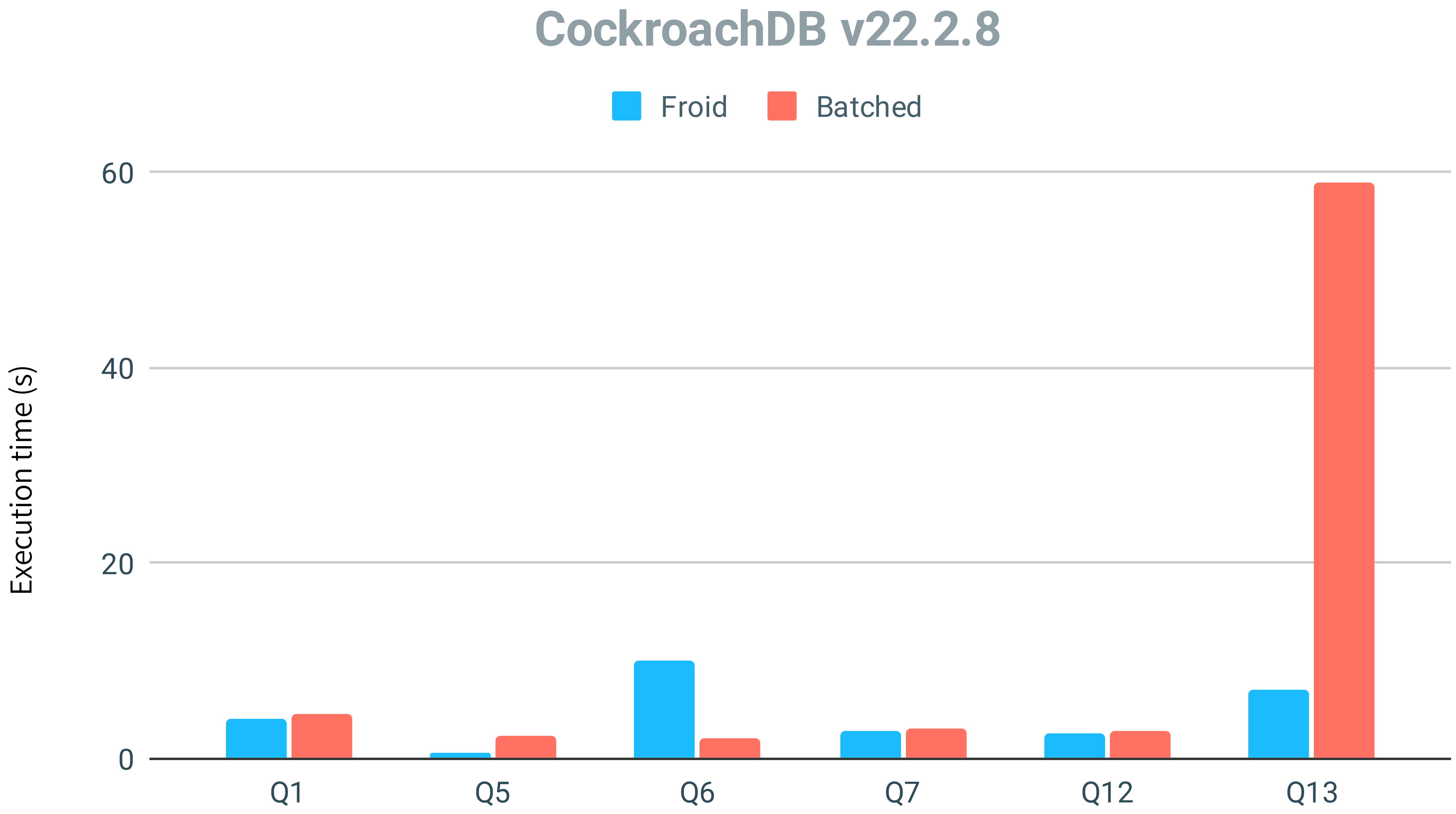
Experiments

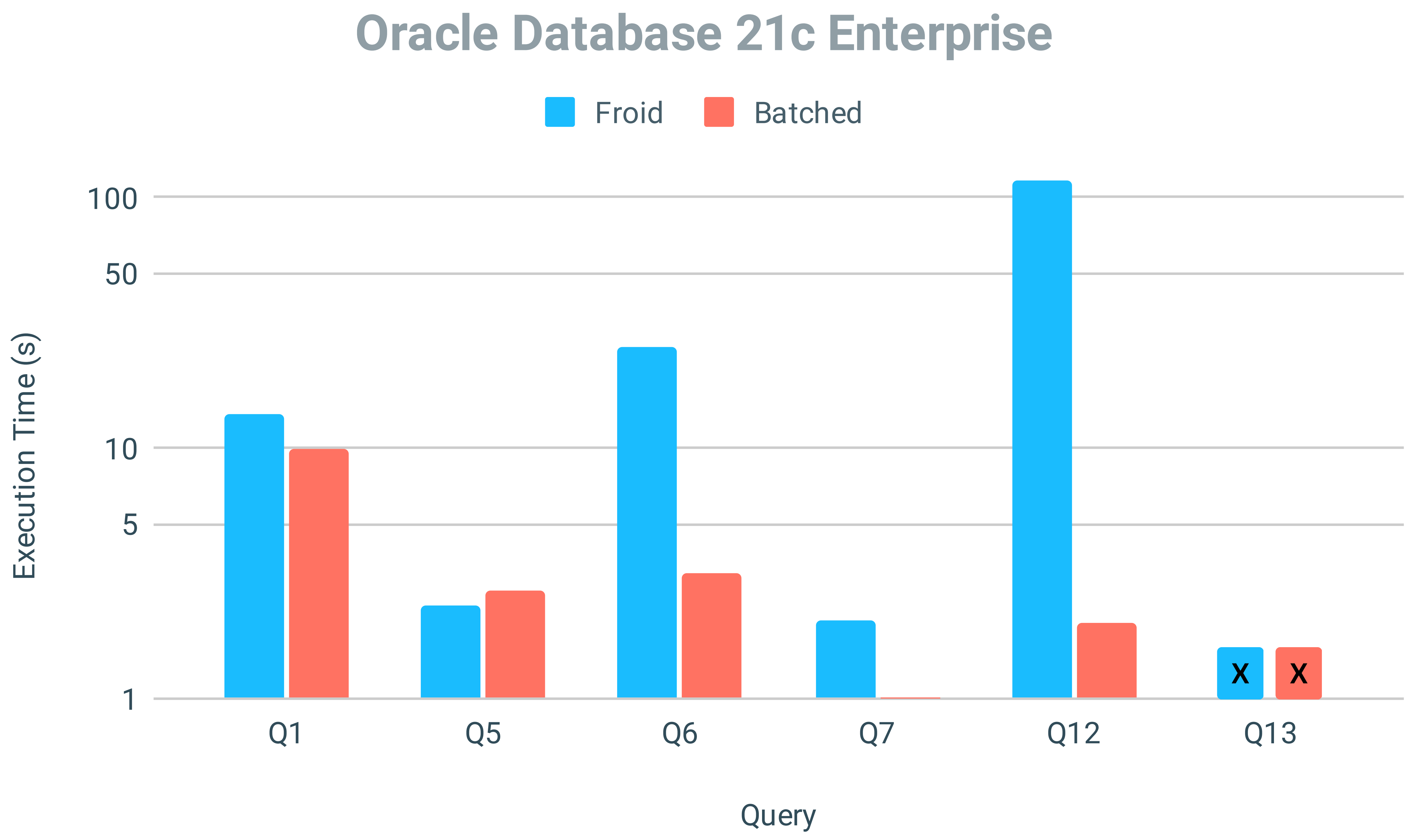
- Postgres not good at decorrelation
- Batching works well with SQL->SQL decorrelation but unsure if this is a fair comparison



Experiments

- CockroachDB is pretty good at handling Froid queries
- Temp tables still an experimental feature



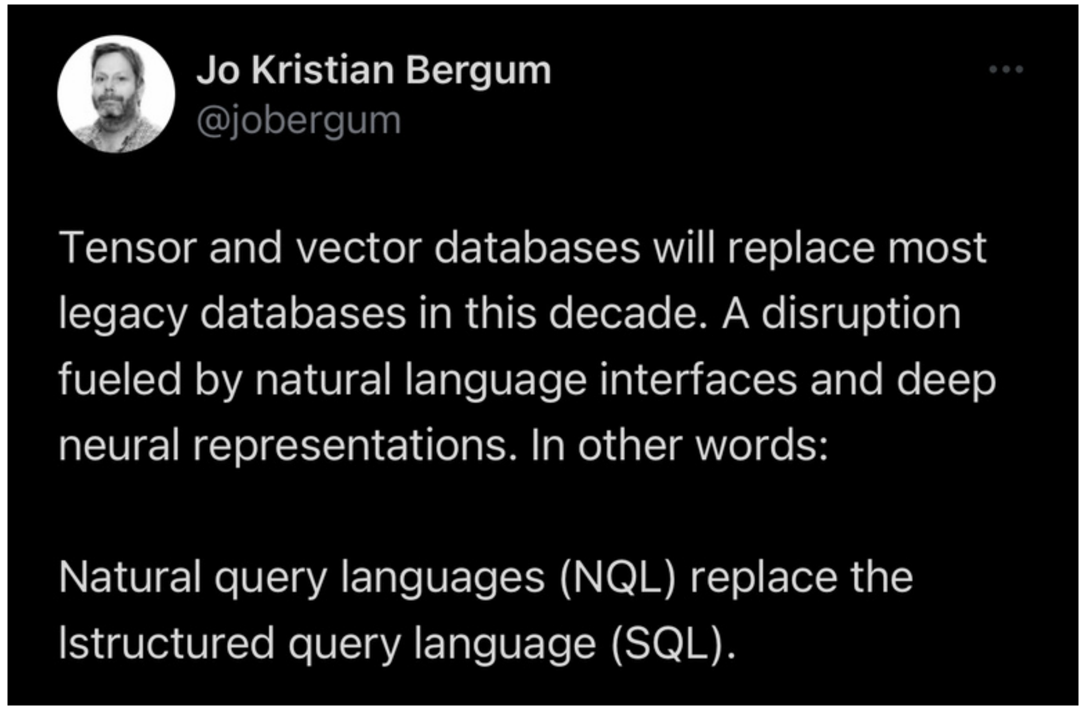


X = timeout (>30 min)

- Very inconsistent with Froid queries
- Batched queries have less variance

Future Work

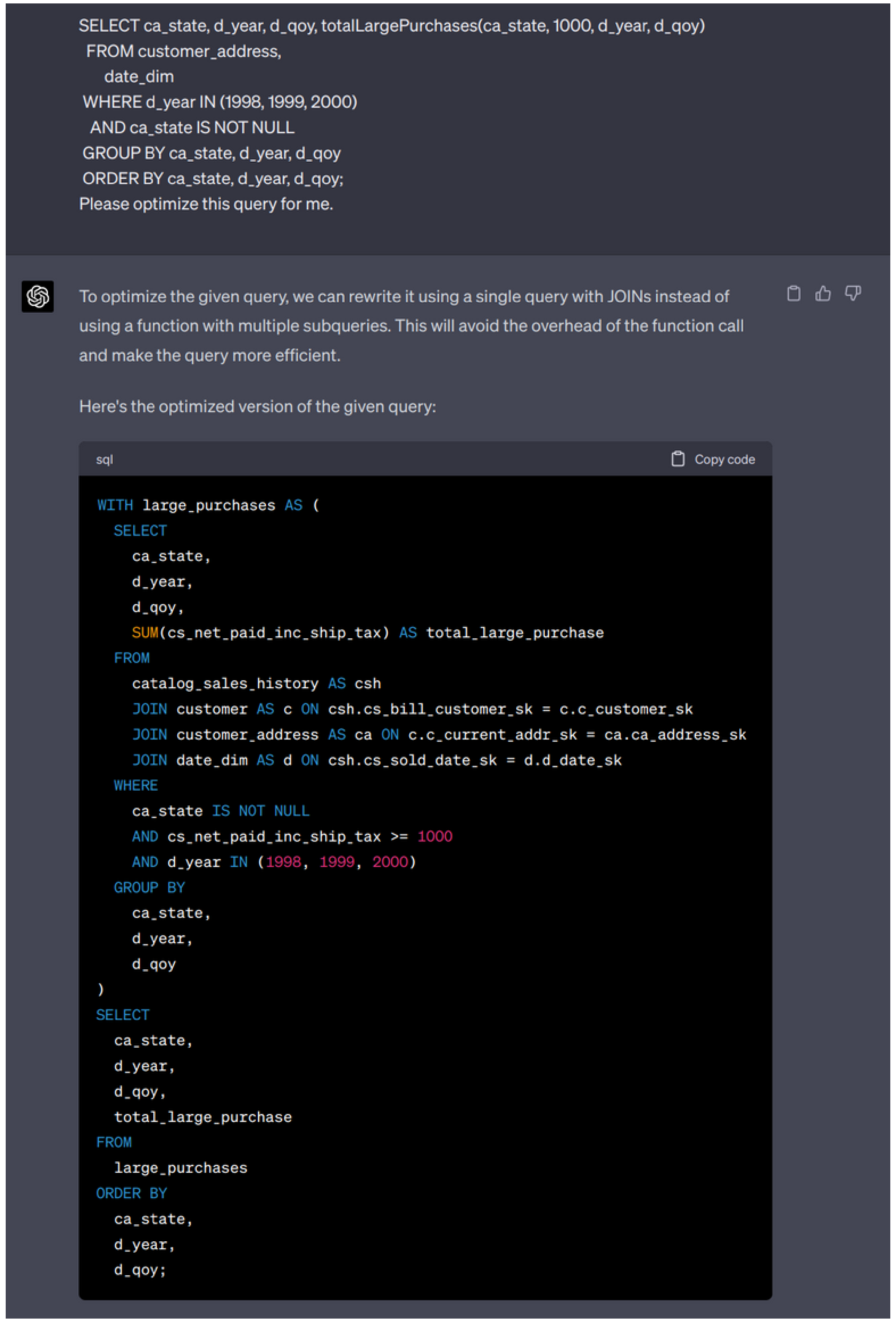
- Test on MySQL, DuckDB
- Optimizing procedural code using large language models



Jo Kristian Bergum
@jobergum

Tensor and vector databases will replace most legacy databases in this decade. A disruption fueled by natural language interfaces and deep neural representations. In other words:

Natural query languages (NQL) replace the Istructured query language (SQL).



```
SELECT ca_state, d_year, d_qoy, totalLargePurchases(ca_state, 1000, d_year, d_qoy)
 FROM customer_address,
    date_dim
 WHERE d_year IN (1998, 1999, 2000)
  AND ca_state IS NOT NULL
 GROUP BY ca_state, d_year, d_qoy
 ORDER BY ca_state, d_year, d_qoy;
Please optimize this query for me.
```

To optimize the given query, we can rewrite it using a single query with JOINs instead of using a function with multiple subqueries. This will avoid the overhead of the function call and make the query more efficient.

Here's the optimized version of the given query:

```sql
WITH large_purchases AS (
  SELECT
    ca_state,
    d_year,
    d_qoy,
    SUM(cs_net_paid_inc_ship_tax) AS total_large_purchase
  FROM
    catalog_sales_history AS csh
    JOIN customer AS c ON csh.cs_bill_customer_sk = c.c_customer_sk
    JOIN customer_address AS ca ON c.c_current_addr_sk = ca.ca_address_sk
    JOIN date_dim AS d ON csh.cs_sold_date_sk = d.d_date_sk
  WHERE
    ca_state IS NOT NULL
    AND cs_net_paid_inc_ship_tax >= 1000
    AND d_year IN (1998, 1999, 2000)
  GROUP BY
    ca_state,
    d_year,
    d_qoy
)
SELECT
  ca_state,
  d_year,
  d_qoy,
  total_large_purchase
FROM
  large_purchases
ORDER BY
  ca_state,
  d_year,
  d_qoy;
```

Thank you!