```
/* ============================================================
BUSINESS CONTEXT: MAVEN FUZZY FACTORY
============================================================

Maven Fuzzy Factory is an e-commerce retailer specializing in
teddy bears. This database captures the full customer journey:
from website sessions and pageviews, through orders, order items,
and refunds.

The data enables analysis of:
• Website traffic growth and seasonality
• Session-to-order conversion performance
• Marketing channel effectiveness
• Revenue efficiency (per order & per session)
• Product-level and refund behavior

The goal of this analysis is to understand traffic trends,
conversion performance, and revenue drivers to help optimize
marketing spend and website performance.
============================================================ */


/* ============================================================
DATABASE SETUP
============================================================ */

-- Create and select database
CREATE DATABASE maven_fuzzy_factory;
USE maven_fuzzy_factory;


/* ============================================================
RAW TABLE CREATION
============================================================ */

-- Stores session-level marketing and device data
CREATE TABLE website_sessions (
   website_session_id INT,
   created_at DATETIME,
   user_id INT,
   utm_source VARCHAR(50),
   utm_campaign VARCHAR(50),
   utm_content VARCHAR(50),
   device_type VARCHAR(20),
   http_referer VARCHAR(255)
);

-- Stores individual pageviews tied to sessions
```

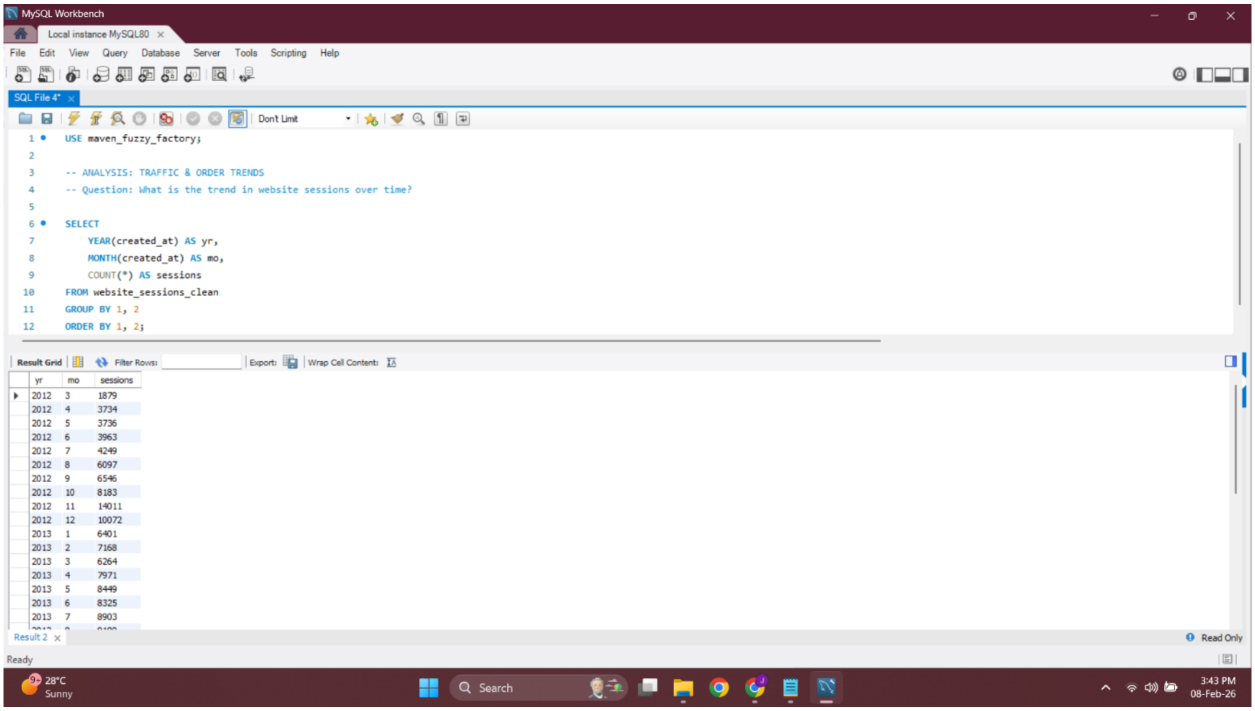
```
CREATE TABLE website_pageviews (
  website_pageview_id INT,
  created_at DATETIME,
  website_session_id INT,
  pageview_url VARCHAR(100)
);

-- Stores completed orders
CREATE TABLE orders (
  order_id INT,
  created_at DATETIME,
  website_session_id INT,
  user_id INT,
  primary_product_id INT,
  items_purchased INT,
  price_usd DECIMAL(10,2),
  cogs_usd DECIMAL(10,2)
);

-- Stores line-item detail for each order
CREATE TABLE order_items (
  order_item_id INT,
  created_at DATETIME,
  order_id INT,
  product_id INT,
  is_primary_item INT,
  price_usd DECIMAL(10,2),
  cogs_usd DECIMAL(10,2)
);

-- Stores refunds issued for order items
CREATE TABLE order_item_refunds (
  order_item_refund_id INT,
  created_at DATETIME,
  order_item_id INT,
  refund_amount_usd DECIMAL(10,2)
);

-- Stores product metadata
CREATE TABLE products (
  product_id INT,
  created_at DATETIME,
  product_name VARCHAR(100)
);


/* ============================================================
DATA LOADING
```

```sql
============================================================ */

-- Enable local file loading
SET GLOBAL local_infile = 1;
USE maven_fuzzy_factory;

-- Load CSV files into raw tables
LOAD DATA LOCAL INFILE
'C:/Users/digvi/Desktop/Sharpener/Portfolio/SQL/Maven+Fuzzy+Factory/website_sessions.csv'
INTO TABLE website_sessions
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 ROWS;

LOAD DATA LOCAL INFILE
'C:/Users/digvi/Desktop/Sharpener/Portfolio/SQL/Maven+Fuzzy+Factory/website_pageviews.csv'
INTO TABLE website_pageviews
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 ROWS;

LOAD DATA LOCAL INFILE
'C:/Users/digvi/Desktop/Sharpener/Portfolio/SQL/Maven+Fuzzy+Factory/orders.csv'
INTO TABLE orders
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 ROWS;

LOAD DATA LOCAL INFILE
'C:/Users/digvi/Desktop/Sharpener/Portfolio/SQL/Maven+Fuzzy+Factory/order_items.csv'
INTO TABLE order_items
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 ROWS;

LOAD DATA LOCAL INFILE
'C:/Users/digvi/Desktop/Sharpener/Portfolio/SQL/Maven+Fuzzy+Factory/order_item_refunds.csv'
INTO TABLE order_item_refunds
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 ROWS;
```

```
LOAD DATA LOCAL INFILE
'C:/Users/digvi/Desktop/Sharpener/Portfolio/SQL/Maven+Fuzzy+Factory/products.csv'
INTO TABLE products
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 ROWS;


/* ==========================================================
DATA VALIDATION & QUALITY CHECKS
========================================================== */

-- Validate row counts after loading
SELECT COUNT(*) FROM website_sessions;
SELECT COUNT(*) FROM website_pageviews;
SELECT COUNT(*) FROM orders;
SELECT COUNT(*) FROM order_items;
SELECT COUNT(*) FROM order_item_refunds;
SELECT COUNT(*) FROM products;

-- Check for nulls in key session fields
SELECT
   COUNT(*) AS total_rows,
   SUM(website_session_id IS NULL) AS null_session_id,
   SUM(created_at IS NULL) AS null_created_at,
   SUM(user_id IS NULL) AS null_user_id
FROM website_sessions;

-- Check for nulls in key order fields
SELECT
   COUNT(*) AS total_rows,
   SUM(order_id IS NULL) AS null_order_id,
   SUM(website_session_id IS NULL) AS null_session_id,
   SUM(price_usd IS NULL) AS null_price
FROM orders;

-- Identify duplicate session records
SELECT website_session_id, COUNT(*)
FROM website_sessions
GROUP BY website_session_id
HAVING COUNT(*) > 1;

-- Identify duplicate order records
SELECT order_id, COUNT(*)
FROM orders
GROUP BY order_id
HAVING COUNT(*) > 1;
```

```
-- Validate date range of sessions
SELECT
   MIN(created_at) AS earliest_date,
   MAX(created_at) AS latest_date
FROM website_sessions;

-- Check for orphan orders without sessions
SELECT COUNT(*) AS orders_without_sessions
FROM orders o
LEFT JOIN website_sessions ws
   ON o.website_session_id = ws.website_session_id
WHERE ws.website_session_id IS NULL;

-- Check for orphan pageviews without sessions
SELECT COUNT(*) AS pageviews_without_sessions
FROM website_pageviews wp
LEFT JOIN website_sessions ws
   ON wp.website_session_id = ws.website_session_id
WHERE ws.website_session_id IS NULL;

-- Identify refunds exceeding original item price
SELECT r.*
FROM order_item_refunds r
JOIN order_items oi
   ON r.order_item_id = oi.order_item_id
WHERE r.refund_amount_usd > oi.price_usd;


/* ==========================================================
DATA CLEANING & STANDARDIZATION
========================================================== */

-- Clean website sessions (remove null identifiers)
CREATE TABLE website_sessions_clean AS
SELECT *
FROM website_sessions
WHERE website_session_id IS NOT NULL
  AND created_at IS NOT NULL;

-- Clean orders (remove null IDs and zero/negative revenue)
CREATE TABLE orders_clean AS
SELECT DISTINCT *
FROM orders
WHERE order_id IS NOT NULL
  AND price_usd > 0;

-- Clean pageviews with valid sessions only
```

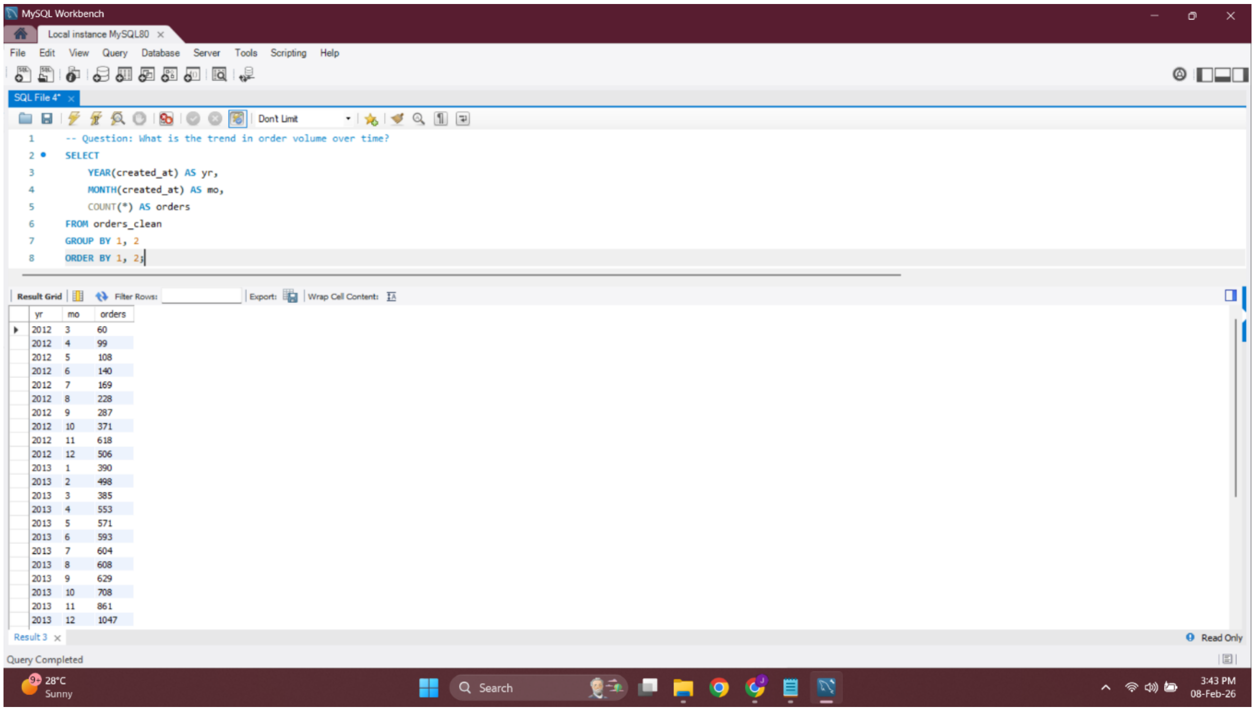
```
CREATE TABLE website_pageviews_clean AS
SELECT DISTINCT wp.*
FROM website_pageviews wp
JOIN website_sessions_clean ws
   ON wp.website_session_id = ws.website_session_id
WHERE wp.website_pageview_id IS NOT NULL
  AND wp.created_at IS NOT NULL;

-- Clean order items with valid orders
CREATE TABLE order_items_clean AS
SELECT DISTINCT oi.*
FROM order_items oi
JOIN orders_clean o
   ON oi.order_id = o.order_id
WHERE oi.order_item_id IS NOT NULL
  AND oi.price_usd IS NOT NULL;

-- Clean refunds and flag over-refunds
CREATE TABLE order_item_refunds_clean AS
SELECT DISTINCT
   r.order_item_refund_id,
   r.created_at,
   r.order_item_id,
   r.refund_amount_usd,
   CASE
      WHEN r.refund_amount_usd > oi.price_usd THEN 1
      ELSE 0
   END AS refund_exceeds_item_price
FROM order_item_refunds r
JOIN order_items_clean oi
   ON r.order_item_id = oi.order_item_id
WHERE r.order_item_refund_id IS NOT NULL;

-- Clean products
CREATE TABLE products_clean AS
SELECT DISTINCT *
FROM products
WHERE product_id IS NOT NULL
  AND product_name IS NOT NULL;


/* ============================================================
REFUND QUALITY SUMMARY
============================================================ */

SELECT
   refund_exceeds_item_price,
   COUNT(*) AS refund_count,
    SUM(refund_amount_usd) AS total_refund_usd
FROM order_item_refunds_clean
GROUP BY refund_exceeds_item_price;


/* ============================================================
CLEAN TABLE ROW COUNTS
============================================================ */

SELECT COUNT(*) FROM website_sessions_clean;
SELECT COUNT(*) FROM website_pageviews_clean;
SELECT COUNT(*) FROM orders_clean;
SELECT COUNT(*) FROM order_items_clean;
SELECT COUNT(*) FROM order_item_refunds_clean;
SELECT COUNT(*) FROM products_clean;


/* ============================================================
ANALYSIS: TRAFFIC & ORDER TRENDS
============================================================ */

-- Question: What is the trend in website sessions over time?
SELECT
    YEAR(created_at) AS yr,
    MONTH(created_at) AS mo,
    COUNT(*) AS sessions
FROM website_sessions_clean
GROUP BY 1, 2
ORDER BY 1, 2;
```



```sql
USE maven_fuzzy_factory;

-- ANALYSIS: TRAFFIC & ORDER TRENDS
-- Question: What is the trend in website sessions over time?

SELECT
    YEAR(created_at) AS yr,
    MONTH(created_at) AS mo,
    COUNT(*) AS sessions
FROM website_sessions_clean
GROUP BY 1, 2
ORDER BY 1, 2;
```

yr	mo	sessions
2012	3	1879
2012	4	3734
2012	5	3736
2012	6	3963
2012	7	4249
2012	8	6097
2012	9	6546
2012	10	8183
2012	11	14011
2012	12	10072
2013	1	6401
2013	2	7168
2013	3	6264
2013	4	7971
2013	5	8449
2013	6	8325
2013	7	8903

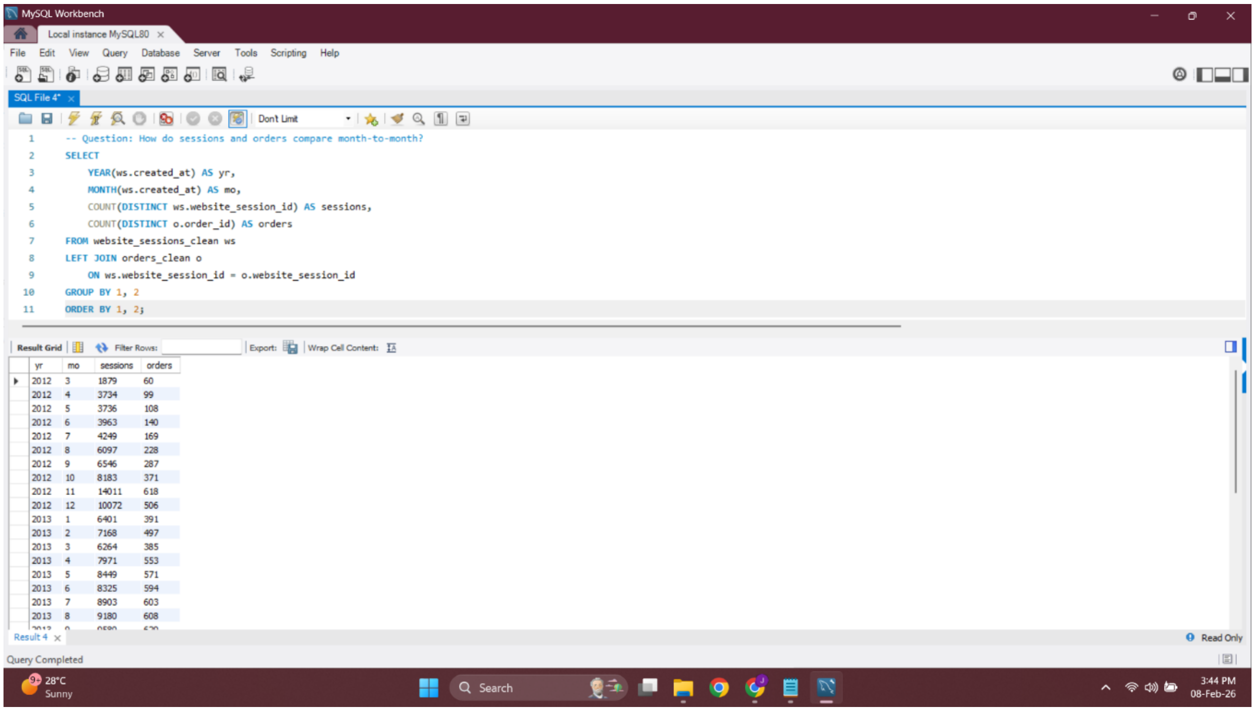
```sql
-- Question: What is the trend in order volume over time?
SELECT
   YEAR(created_at) AS yr,
   MONTH(created_at) AS mo,
   COUNT(*) AS orders
FROM orders_clean
GROUP BY 1, 2
ORDER BY 1, 2;
```



```sql
-- Question: What is the trend in order volume over time?
SELECT
    YEAR(created_at) AS yr,
    MONTH(created_at) AS mo,
    COUNT(*) AS orders
FROM orders_clean
GROUP BY 1, 2
ORDER BY 1, 2;
```

yr	mo	orders
2012	3	60
2012	4	99
2012	5	108
2012	6	140
2012	7	169
2012	8	228
2012	9	287
2012	10	371
2012	11	618
2012	12	506
2013	1	390
2013	2	498
2013	3	385
2013	4	553
2013	5	571
2013	6	593
2013	7	604
2013	8	608
2013	9	629
2013	10	708
2013	11	861
2013	12	1047

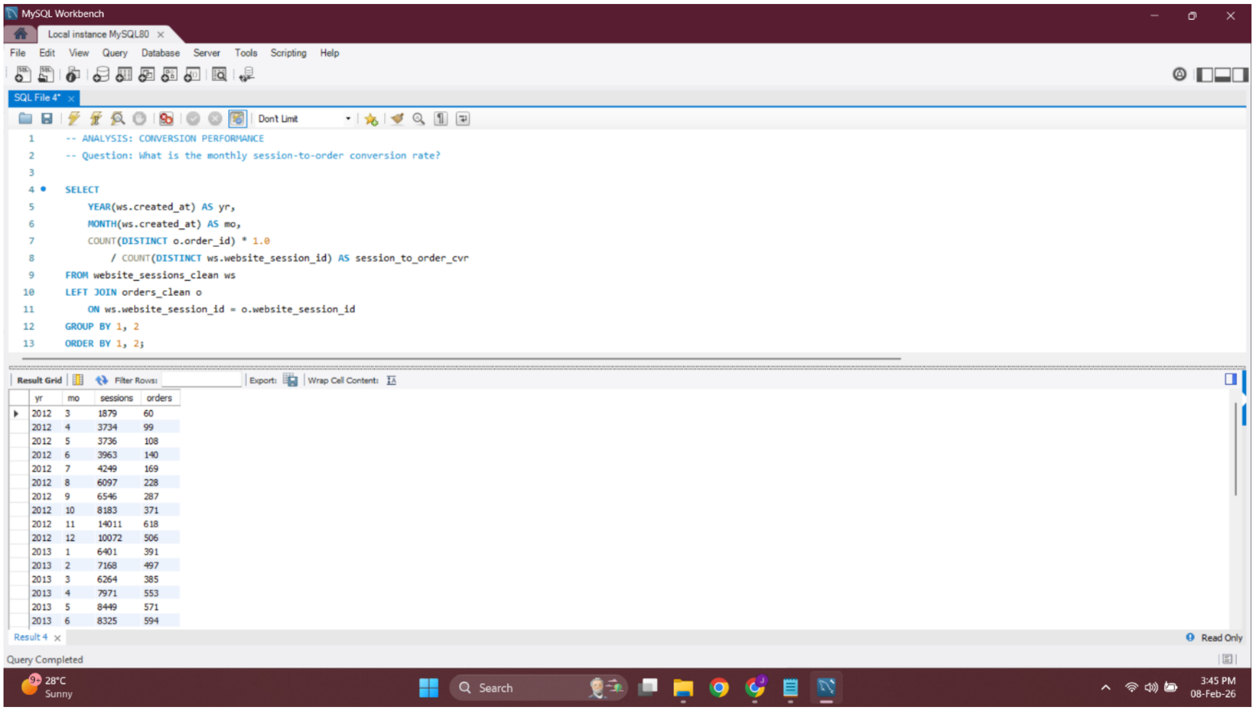
```sql
-- Question: How do sessions and orders compare month-to-month?
SELECT
  YEAR(ws.created_at) AS yr,
  MONTH(ws.created_at) AS mo,
  COUNT(DISTINCT ws.website_session_id) AS sessions,
  COUNT(DISTINCT o.order_id) AS orders
FROM website_sessions_clean ws
LEFT JOIN orders_clean o
  ON ws.website_session_id = o.website_session_id
GROUP BY 1, 2
ORDER BY 1, 2;
```



```sql
-- Question: How do sessions and orders compare month-to-month?
SELECT
    YEAR(ws.created_at) AS yr,
    MONTH(ws.created_at) AS mo,
    COUNT(DISTINCT ws.website_session_id) AS sessions,
    COUNT(DISTINCT o.order_id) AS orders
FROM website_sessions_clean ws
LEFT JOIN orders_clean o
    ON ws.website_session_id = o.website_session_id
GROUP BY 1, 2
ORDER BY 1, 2;
```

yr	mo	sessions	orders
2012	3	1879	60
2012	4	3734	99
2012	5	3736	108
2012	6	3963	140
2012	7	4249	169
2012	8	6097	228
2012	9	6546	287
2012	10	8183	371
2012	11	14011	618
2012	12	10072	506
2013	1	6401	391
2013	2	7168	497
2013	3	6264	385
2013	4	7971	553
2013	5	8449	571
2013	6	8325	594
2013	7	8903	603
2013	8	9180	608

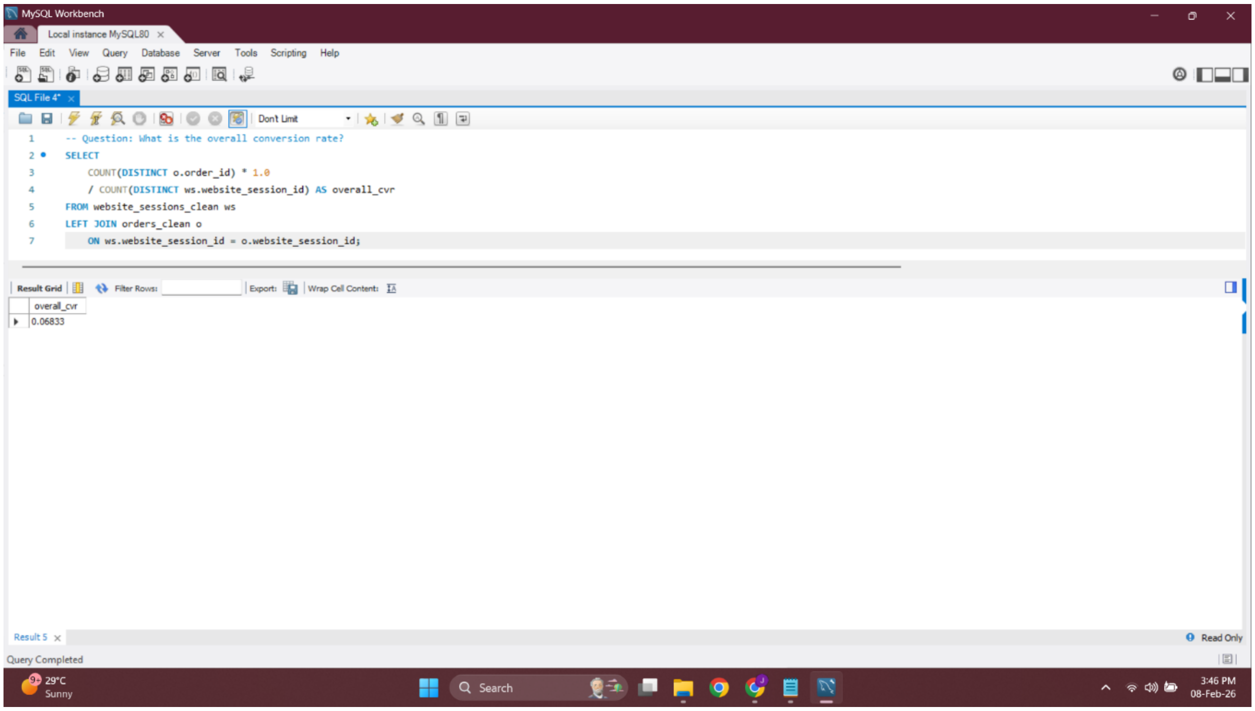
```
/* ============================================================
ANALYSIS: CONVERSION PERFORMANCE
============================================================ */

-- Question: What is the monthly session-to-order conversion rate?
SELECT
    YEAR(ws.created_at) AS yr,
    MONTH(ws.created_at) AS mo,
    COUNT(DISTINCT o.order_id) * 1.0
        / COUNT(DISTINCT ws.website_session_id) AS session_to_order_cvr
FROM website_sessions_clean ws
LEFT JOIN orders_clean o
    ON ws.website_session_id = o.website_session_id
GROUP BY 1, 2
ORDER BY 1, 2;
```



```sql
-- ANALYSIS: CONVERSION PERFORMANCE
-- Question: What is the monthly session-to-order conversion rate?

SELECT
    YEAR(ws.created_at) AS yr,
    MONTH(ws.created_at) AS mo,
    COUNT(DISTINCT o.order_id) * 1.0
        / COUNT(DISTINCT ws.website_session_id) AS session_to_order_cvr
FROM website_sessions_clean ws
LEFT JOIN orders_clean o
    ON ws.website_session_id = o.website_session_id
GROUP BY 1, 2
ORDER BY 1, 2;
```

yr	mo	sessions	orders
2012	3	1879	60
2012	4	3734	99
2012	5	3736	108
2012	6	3963	140
2012	7	4249	169
2012	8	6097	228
2012	9	6546	287
2012	10	8183	371
2012	11	14011	618
2012	12	10072	506
2013	1	6401	391
2013	2	7168	497
2013	3	6264	385
2013	4	7971	553
2013	5	8449	571
2013	6	8325	594

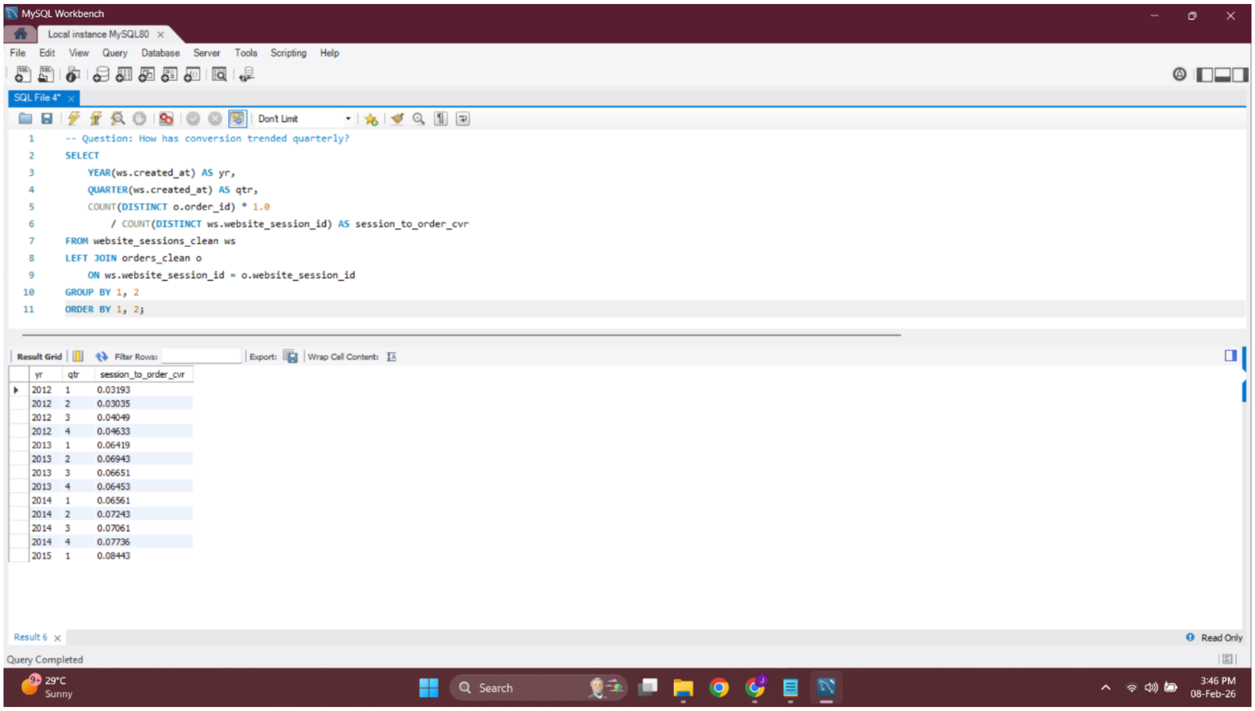
```sql
-- Question: What is the overall conversion rate?
SELECT
    COUNT(DISTINCT o.order_id) * 1.0
    / COUNT(DISTINCT ws.website_session_id) AS overall_cvr
FROM website_sessions_clean ws
LEFT JOIN orders_clean o
    ON ws.website_session_id = o.website_session_id;
```



```sql
-- Question: What is the overall conversion rate?
SELECT
    COUNT(DISTINCT o.order_id) * 1.0
    / COUNT(DISTINCT ws.website_session_id) AS overall_cvr
FROM website_sessions_clean ws
LEFT JOIN orders_clean o
    ON ws.website_session_id = o.website_session_id;
```

```
overall_cvr
0.06833
```

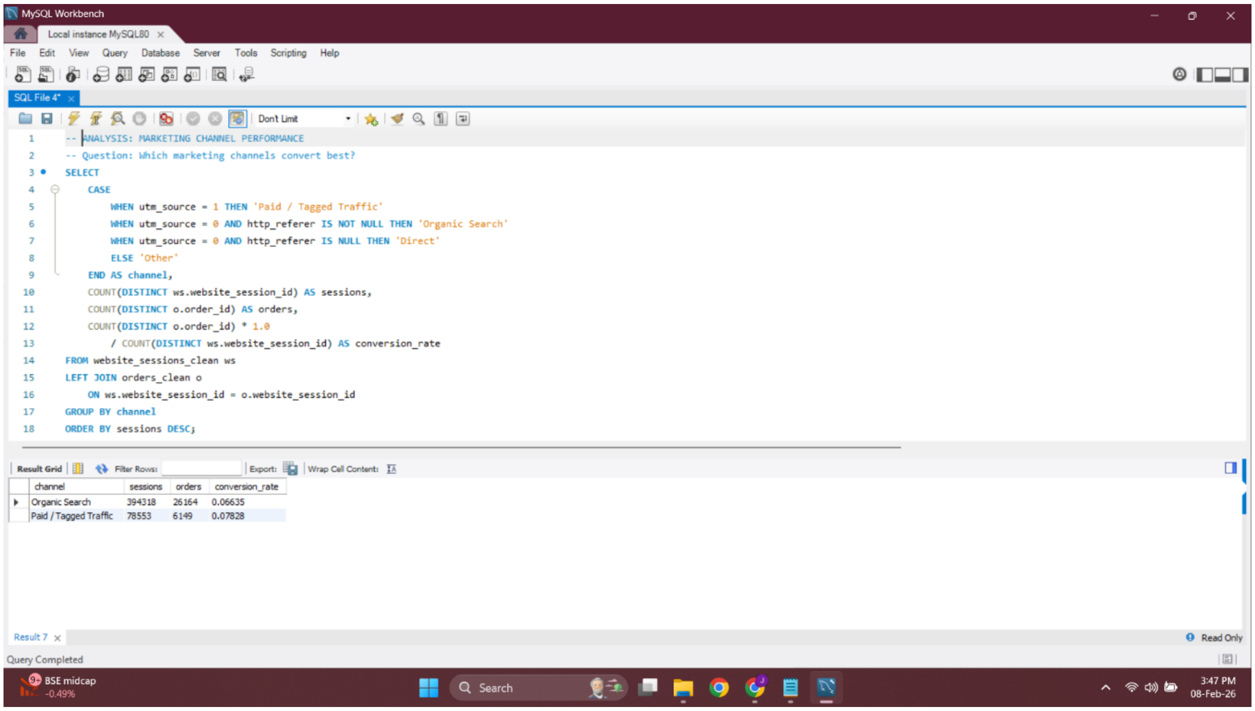
```sql
-- Question: How has conversion trended quarterly?
SELECT
   YEAR(ws.created_at) AS yr,
   QUARTER(ws.created_at) AS qtr,
   COUNT(DISTINCT o.order_id) * 1.0
      / COUNT(DISTINCT ws.website_session_id) AS session_to_order_cvr
FROM website_sessions_clean ws
LEFT JOIN orders_clean o
   ON ws.website_session_id = o.website_session_id
GROUP BY 1, 2
ORDER BY 1, 2;
```



```
-- Question: How has conversion trended quarterly?
SELECT
    YEAR(ws.created_at) AS yr,
    QUARTER(ws.created_at) AS qtr,
    COUNT(DISTINCT o.order_id) * 1.0
        / COUNT(DISTINCT ws.website_session_id) AS session_to_order_cvr
FROM website_sessions_clean ws
LEFT JOIN orders_clean o
    ON ws.website_session_id = o.website_session_id
GROUP BY 1, 2
ORDER BY 1, 2;
```

yr	qtr	session_to_order_cvr
2012	1	0.03193
2012	2	0.03035
2012	3	0.04049
2012	4	0.04633
2013	1	0.06419
2013	2	0.06943
2013	3	0.06651
2013	4	0.06453
2014	1	0.06561
2014	2	0.07243
2014	3	0.07061
2014	4	0.07736
2015	1	0.08443

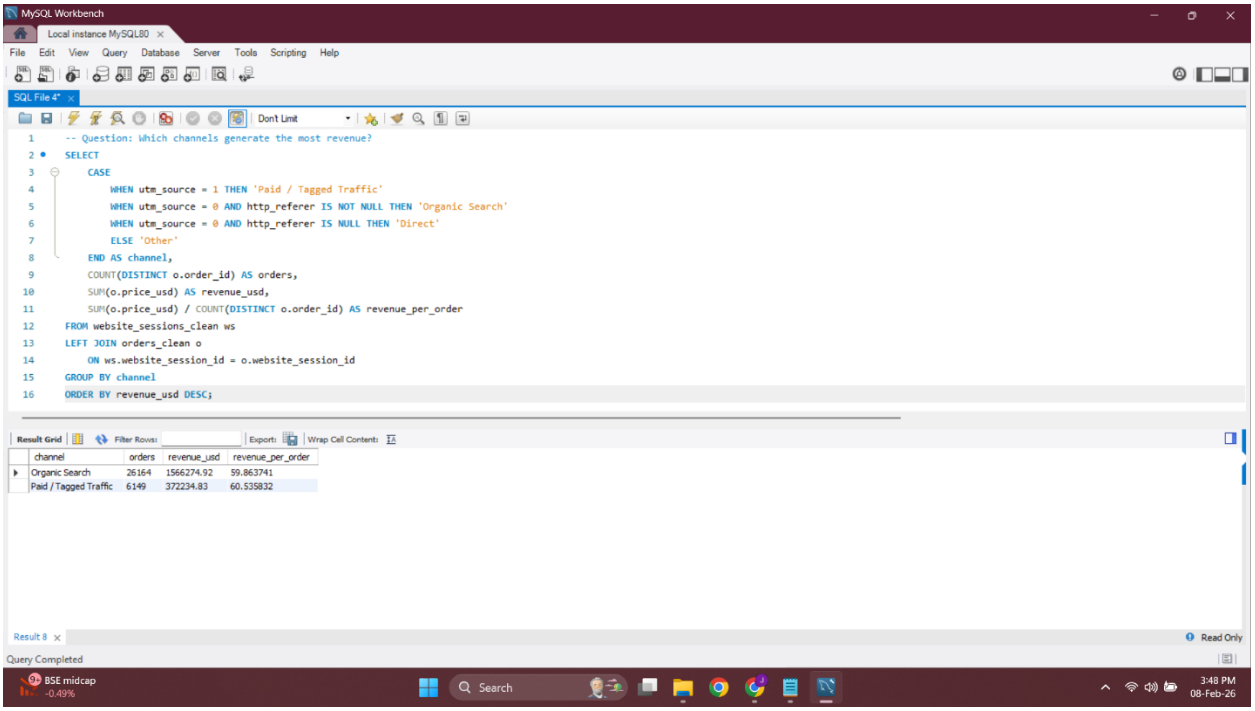
```
/* ============================================================
MARKETING CHANNEL NORMALIZATION
============================================================ */

-- Normalize empty UTM values
SET SQL_SAFE_UPDATES = 0;
UPDATE website_sessions_clean
SET utm_source = NULL
WHERE utm_source = '';
SET SQL_SAFE_UPDATES = 1;


/* ============================================================
ANALYSIS: MARKETING CHANNEL PERFORMANCE
============================================================ */

-- Question: Which marketing channels convert best?
SELECT
   CASE
      WHEN utm_source = 1 THEN 'Paid / Tagged Traffic'
      WHEN utm_source = 0 AND http_referer IS NOT NULL THEN 'Organic Search'
      WHEN utm_source = 0 AND http_referer IS NULL THEN 'Direct'
      ELSE 'Other'
   END AS channel,
   COUNT(DISTINCT ws.website_session_id) AS sessions,
   COUNT(DISTINCT o.order_id) AS orders,
  COUNT(DISTINCT o.order_id) * 1.0
    / COUNT(DISTINCT ws.website_session_id) AS conversion_rate
FROM website_sessions_clean ws
LEFT JOIN orders_clean o
  ON ws.website_session_id = o.website_session_id
GROUP BY channel
ORDER BY sessions DESC;
```



```
-- ANALYSIS: MARKETING CHANNEL PERFORMANCE
-- Question: Which marketing channels convert best?
SELECT
    CASE
        WHEN utm_source = 1 THEN 'Paid / Tagged Traffic'
        WHEN utm_source = 0 AND http_referer IS NOT NULL THEN 'Organic Search'
        WHEN utm_source = 0 AND http_referer IS NULL THEN 'Direct'
        ELSE 'Other'
    END AS channel,
    COUNT(DISTINCT ws.website_session_id) AS sessions,
    COUNT(DISTINCT o.order_id) AS orders,
    COUNT(DISTINCT o.order_id) * 1.0
        / COUNT(DISTINCT ws.website_session_id) AS conversion_rate
FROM website_sessions_clean ws
LEFT JOIN orders_clean o
    ON ws.website_session_id = o.website_session_id
GROUP BY channel
ORDER BY sessions DESC;
```

channel	sessions	orders	conversion_rate
Organic Search	394318	26164	0.06635
Paid / Tagged Traffic	78553	6149	0.07828

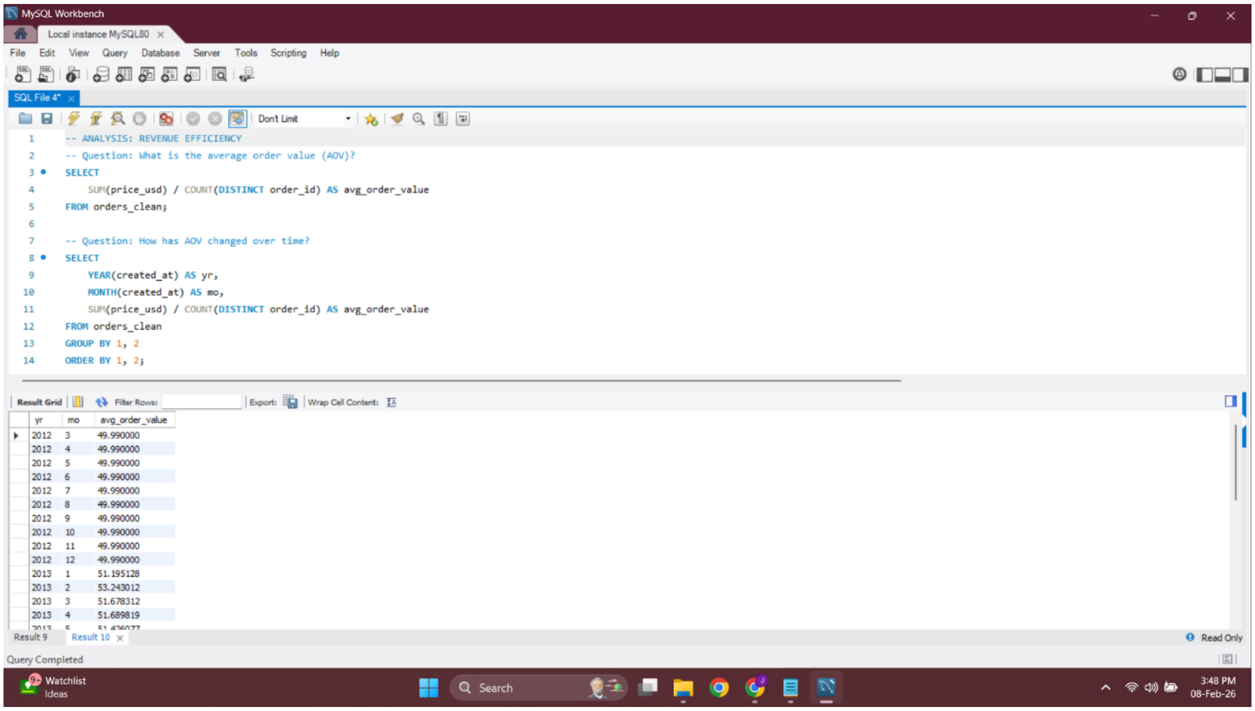
```
-- Question: Which channels generate the most revenue?
SELECT
  CASE
    WHEN utm_source = 1 THEN 'Paid / Tagged Traffic'
    WHEN utm_source = 0 AND http_referer IS NOT NULL THEN 'Organic Search'
    WHEN utm_source = 0 AND http_referer IS NULL THEN 'Direct'
    ELSE 'Other'
  END AS channel,
  COUNT(DISTINCT o.order_id) AS orders,
  SUM(o.price_usd) AS revenue_usd,
  SUM(o.price_usd) / COUNT(DISTINCT o.order_id) AS revenue_per_order
FROM website_sessions_clean ws
LEFT JOIN orders_clean o
  ON ws.website_session_id = o.website_session_id
GROUP BY channel
ORDER BY revenue_usd DESC;
```



```
-- Question: Which channels generate the most revenue?
SELECT
    CASE
        WHEN utm_source = 1 THEN 'Paid / Tagged Traffic'
        WHEN utm_source = 0 AND http_referer IS NOT NULL THEN 'Organic Search'
        WHEN utm_source = 0 AND http_referer IS NULL THEN 'Direct'
        ELSE 'Other'
    END AS channel,
    COUNT(DISTINCT o.order_id) AS orders,
    SUM(o.price_usd) AS revenue_usd,
    SUM(o.price_usd) / COUNT(DISTINCT o.order_id) AS revenue_per_order
FROM website_sessions_clean ws
LEFT JOIN orders_clean o
    ON ws.website_session_id = o.website_session_id
GROUP BY channel
ORDER BY revenue_usd DESC;
```

channel	orders	revenue_usd	revenue_per_order
Organic Search	26164	1566274.92	59.863741
Paid / Tagged Traffic	6149	372234.83	60.535832

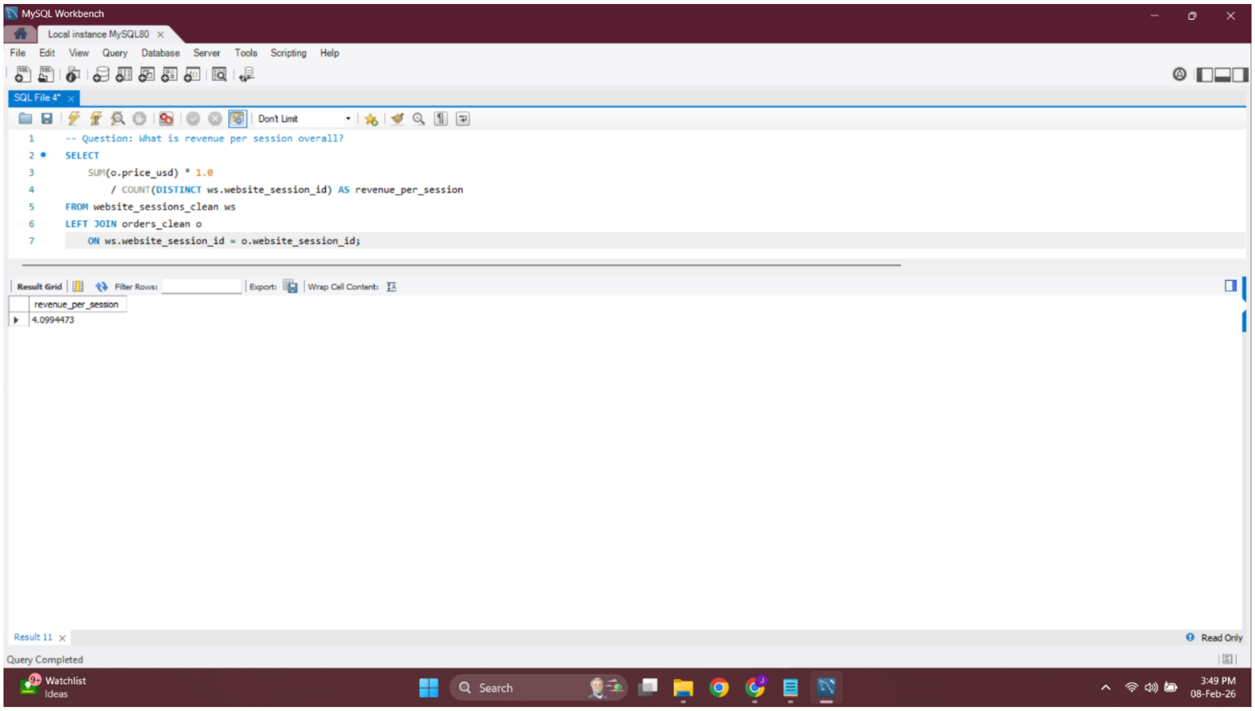
```
/* ============================================================
ANALYSIS: REVENUE EFFICIENCY
============================================================ */

-- Question: What is the average order value (AOV)?
SELECT
   SUM(price_usd) / COUNT(DISTINCT order_id) AS avg_order_value
FROM orders_clean;

-- Question: How has AOV changed over time?
SELECT
   YEAR(created_at) AS yr,
   MONTH(created_at) AS mo,
   SUM(price_usd) / COUNT(DISTINCT order_id) AS avg_order_value
FROM orders_clean
GROUP BY 1, 2
ORDER BY 1, 2;
```



```sql
-- ANALYSIS: REVENUE EFFICIENCY
-- Question: What is the average order value (AOV)?
SELECT
    SUM(price_usd) / COUNT(DISTINCT order_id) AS avg_order_value
FROM orders_clean;

-- Question: How has AOV changed over time?
SELECT
    YEAR(created_at) AS yr,
    MONTH(created_at) AS mo,
    SUM(price_usd) / COUNT(DISTINCT order_id) AS avg_order_value
FROM orders_clean
GROUP BY 1, 2
ORDER BY 1, 2;
```

yr	mo	avg_order_value
2012	3	49.990000
2012	4	49.990000
2012	5	49.990000
2012	6	49.990000
2012	7	49.990000
2012	8	49.990000
2012	9	49.990000
2012	10	49.990000
2012	11	49.990000
2012	12	49.990000
2013	1	51.195128
2013	2	53.243012
2013	3	51.678312
2013	4	51.689819
2013	5	51.426077

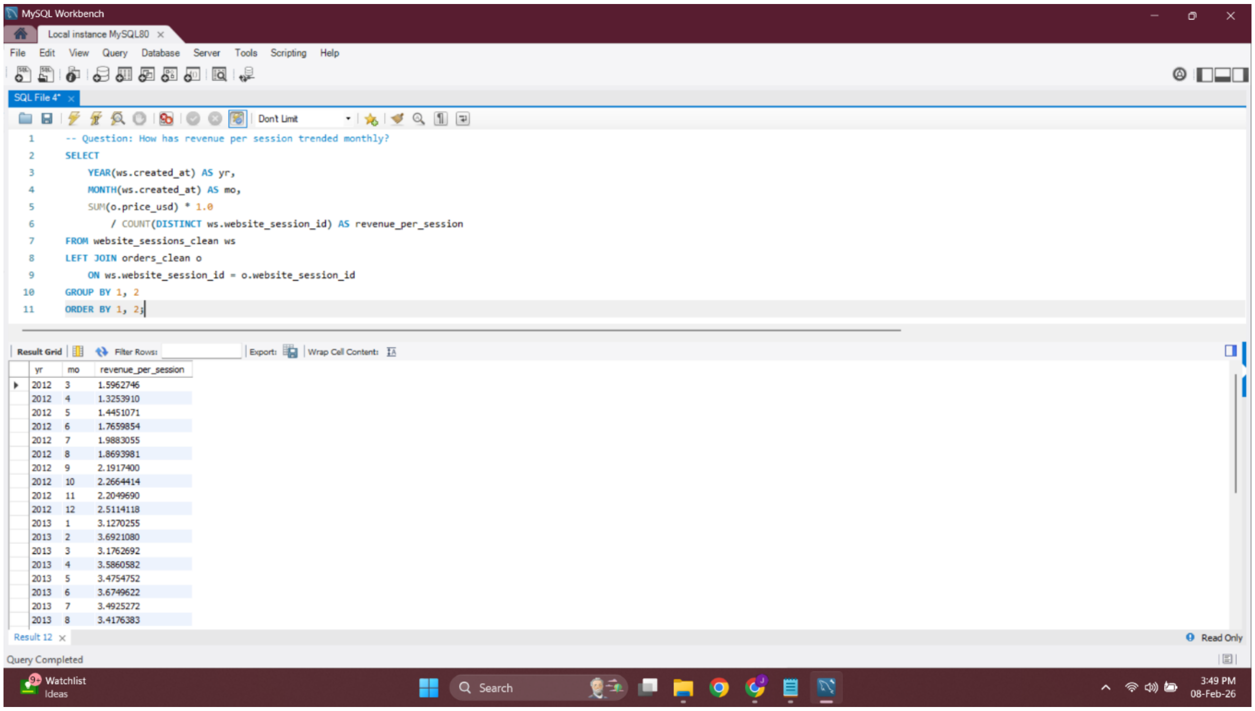
```sql
-- Question: What is revenue per session overall?
SELECT
   SUM(o.price_usd) * 1.0
      / COUNT(DISTINCT ws.website_session_id) AS revenue_per_session
FROM website_sessions_clean ws
LEFT JOIN orders_clean o
   ON ws.website_session_id = o.website_session_id;
```



```
-- Question: What is revenue per session overall?
SELECT
    SUM(o.price_usd) * 1.0
        / COUNT(DISTINCT ws.website_session_id) AS revenue_per_session
FROM website_sessions_clean ws
LEFT JOIN orders_clean o
    ON ws.website_session_id = o.website_session_id;
```

```
revenue_per_session
4.0994473
```

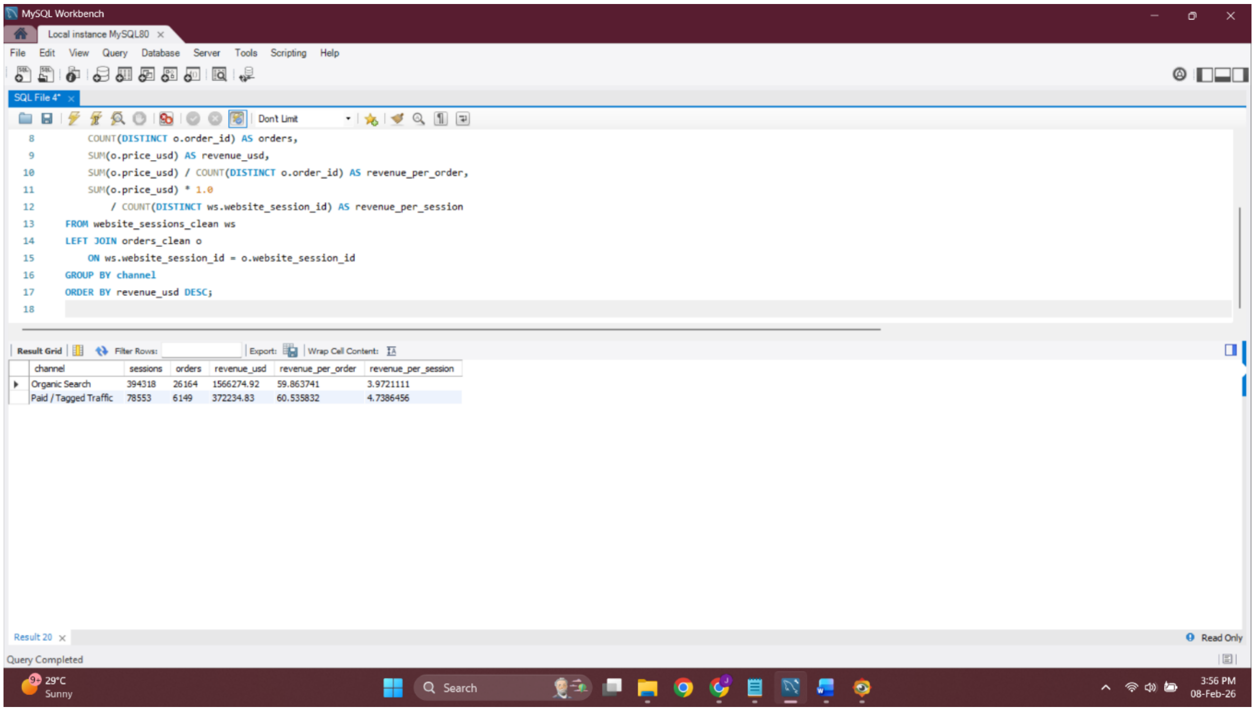
```
-- Question: How has revenue per session trended monthly?
SELECT
    YEAR(ws.created_at) AS yr,
    MONTH(ws.created_at) AS mo,
    SUM(o.price_usd) * 1.0
        / COUNT(DISTINCT ws.website_session_id) AS revenue_per_session
FROM website_sessions_clean ws
LEFT JOIN orders_clean o
    ON ws.website_session_id = o.website_session_id
GROUP BY 1, 2
ORDER BY 1, 2;
```



```
-- Question: How has revenue per session trended monthly?
SELECT
    YEAR(ws.created_at) AS yr,
    MONTH(ws.created_at) AS mo,
    SUM(o.price_usd) * 1.0
        / COUNT(DISTINCT ws.website_session_id) AS revenue_per_session
FROM website_sessions_clean ws
LEFT JOIN orders_clean o
    ON ws.website_session_id = o.website_session_id
GROUP BY 1, 2
ORDER BY 1, 2;
```

yr	mo	revenue_per_session
2012	3	1.5962746
2012	4	1.3253910
2012	5	1.4451071
2012	6	1.7659854
2012	7	1.9883055
2012	8	1.8693981
2012	9	2.1917400
2012	10	2.2664414
2012	11	2.2049690
2012	12	2.5114118
2013	1	3.1270255
2013	2	3.6921080
2013	3	3.1762692
2013	4	3.5860582
2013	5	3.4754752
2013	6	3.6749622
2013	7	3.4925272
2013	8	3.4176383

```
/* ============================================================
FINAL CHANNEL SUMMARY
============================================================ */

SELECT
  CASE
    WHEN utm_source = 1 THEN 'Paid / Tagged Traffic'
    WHEN utm_source = 0 THEN 'Organic Search'
    ELSE 'Other'
  END AS channel,
  COUNT(DISTINCT ws.website_session_id) AS sessions,
  COUNT(DISTINCT o.order_id) AS orders,
  SUM(o.price_usd) AS revenue_usd,
  SUM(o.price_usd) / COUNT(DISTINCT o.order_id) AS revenue_per_order,
  SUM(o.price_usd) * 1.0
    / COUNT(DISTINCT ws.website_session_id) AS revenue_per_session
FROM website_sessions_clean ws
LEFT JOIN orders_clean o
  ON ws.website_session_id = o.website_session_id
GROUP BY channel
ORDER BY revenue_usd DESC;
```



```sql
        COUNT(DISTINCT o.order_id) AS orders,
        SUM(o.price_usd) AS revenue_usd,
        SUM(o.price_usd) / COUNT(DISTINCT o.order_id) AS revenue_per_order,
        SUM(o.price_usd) * 1.0
            / COUNT(DISTINCT ws.website_session_id) AS revenue_per_session
    FROM website_sessions_clean ws
    LEFT JOIN orders_clean o
        ON ws.website_session_id = o.website_session_id
    GROUP BY channel
    ORDER BY revenue_usd DESC;
```

channel	sessions	orders	revenue_usd	revenue_per_order	revenue_per_session
Organic Search	394318	26164	1566274.92	59.863741	3.9721111
Paid / Tagged Traffic	78553	6149	372234.83	60.535832	4.7386456

Recommendations:

1. Reallocate Marketing Spend Toward High-Efficiency Channels
The channel analysis compares sessions, orders, conversion rate, revenue per order, and revenue per session. Use these results to **prioritize channels that deliver higher revenue per session and stronger conversion**, not just higher traffic.
Action: Shift budget away from low-converting paid traffic and double down on channels (often Organic or Direct) that show stronger efficiency and profitability. This will improve overall ROI without necessarily increasing spend.
SQL

---

2. Focus Conversion Rate Optimization During Low-Conversion Periods
Monthly and quarterly conversion trends show that performance is not consistent over time. Rather than treating conversion as static, **identify months or quarters with below-average CVR** and investigate UX, pricing, or messaging issues.
Action: Run targeted A/B tests (landing pages, checkout flow, promotions) during historically weaker periods to stabilize and lift overall conversion.
SQL

---

3. Improve Revenue Per Session Through Upsell & Bundling
Revenue per session and AOV trends indicate how effectively traffic is being monetized. Since order-level data exists, there is an opportunity to **increase basket size rather than only driving more traffic**.
Action: Introduce product bundles, cross-sell teddy bear accessories, or limited-time offers to raise AOV and revenue per session without increasing acquisition costs.
SQL

4. Investigate and Control Refund Anomalies

The refund quality checks explicitly flag cases where refund amounts exceed item prices. Even if rare, these cases represent **data integrity issues and potential revenue leakage**.

Action: Audit the root cause of over-refunds (system errors, manual overrides, or policy gaps) and implement validation rules to prevent refunds greater than original item value.

SQL

5. Standardize and Strengthen Marketing Attribution Logic

Channel classification currently relies on simplified UTM and referrer logic, which can misclassify traffic (e.g., paid vs organic vs direct). This directly impacts marketing decisions.

Action: Improve UTM tagging standards and refine attribution rules so performance reporting is more accurate. Better attribution will lead to smarter budget allocation and clearer channel accountability.