**🧍‍♂️ Customer 360° Analytics**

1. **Customer Profile & Segmentation**

Data\_sources\_required: customers, sales, products, orders, customer\_interactions, customer\_support, email\_campaigns, returns, loyalty\_program, referrals

-- Customer demographics by location, age, gender

SELECT country, state, city, gender, AVG(age) AS avg\_age, COUNT(\*) AS customer\_count

FROM customers

GROUP BY country, state, city, gender;

-- Customer segmentation

SELECT customer\_segment, COUNT(\*) AS num\_customers

FROM customers

GROUP BY customer\_segment;

-- Lifecycle stage

SELECT lifecycle\_stage, COUNT(\*) AS count

FROM customers

GROUP BY lifecycle\_stage;

1. **Customer Lifetime Value (CLV) & Profitability**

-- CLV by acquisition channel

SELECT acquisition\_channel, AVG(clv\_score) AS avg\_clv

FROM customers

GROUP BY acquisition\_channel;

-- High vs low value customers

SELECT

CASE

WHEN clv\_score >= 5000 THEN 'High Value'

WHEN clv\_score >= 1000 THEN 'Medium Value'

ELSE 'Low Value'

END AS value\_segment,

COUNT(\*) AS customer\_count

FROM customers

GROUP BY value\_segment;

-- Net margin per customer

SELECT c.customer\_id, c.first\_name, SUM(s.total\_price - p.cost \* s.quantity) AS net\_margin

FROM sales s

JOIN products p ON s.product\_id = p.product\_id

JOIN orders o ON s.order\_id = o.order\_id

JOIN customers c ON o.customer\_id = c.customer\_id

GROUP BY c.customer\_id;

1. **Behavior & Engagement Analytics**

-- Interaction by device and channel

SELECT device\_type, channel, COUNT(\*) AS interactions

FROM customer\_interactions

GROUP BY device\_type, channel;

-- Email engagement by cohort

SELECT strftime('%Y-%m', sent\_date) AS cohort, COUNT(\*) AS emails\_sent,

SUM(opened) AS opened, SUM(clicked) AS clicked

FROM email\_campaigns

GROUP BY cohort;

-- Product view to purchase conversion

SELECT product\_id,

COUNT(DISTINCT CASE WHEN interaction\_type = 'Page View' THEN session\_id END) AS views,

COUNT(DISTINCT CASE WHEN interaction\_type = 'Purchase' THEN session\_id END) AS purchases,

ROUND(1.0 \* COUNT(DISTINCT CASE WHEN interaction\_type = 'Purchase' THEN session\_id END) /

NULLIF(COUNT(DISTINCT CASE WHEN interaction\_type = 'Page View' THEN session\_id END), 0), 2) AS conversion\_rate

FROM customer\_interactions

GROUP BY product\_id;

1. **Churn & Retention Insights**

-- Churn risk by segment

SELECT customer\_segment, AVG(churn\_risk\_score) AS avg\_churn\_risk

FROM customers

GROUP BY customer\_segment;

-- Retention by acquisition source

SELECT acquisition\_channel,

COUNT(\*) AS total\_customers,

SUM(CASE WHEN is\_active = 1 THEN 1 ELSE 0 END) AS retained\_customers,

ROUND(100.0 \* SUM(CASE WHEN is\_active = 1 THEN 1 ELSE 0 END) / COUNT(\*), 2) AS retention\_rate

FROM customers

GROUP BY acquisition\_channel;

1. **Customer Support & Experience Metrics**

-- Ticket resolution and volume

SELECT agent\_id, COUNT(\*) AS tickets\_handled,

AVG(resolution\_time\_hours) AS avg\_resolution\_time

FROM customer\_support

GROUP BY agent\_id;

-- CSAT trend

SELECT strftime('%Y-%m', created\_date) AS month, AVG(satisfaction\_score) AS avg\_csat

FROM customer\_support

WHERE satisfaction\_score IS NOT NULL

GROUP BY month;

-- Returns/refunds per customer

SELECT customer\_id, COUNT(\*) AS total\_returns, SUM(refund\_amount) AS total\_refunded

FROM returns

GROUP BY customer\_id;

1. **Loyalty & Referral Program Performance**

-- Loyalty tier distribution

SELECT tier, COUNT(\*) AS customers\_in\_tier

FROM loyalty\_program

GROUP BY tier;

-- Referral acquisition

SELECT r.status, COUNT(\*) AS total\_referrals

FROM referrals r

GROUP BY r.status;

-- Rewards redemption

SELECT tier, SUM(points\_earned) AS total\_earned, SUM(points\_redeemed) AS total\_redeemed

FROM loyalty\_program

GROUP BY tier;

**💸 Sales Performance Dashboard**

Data\_sources\_required: customers, sales, products, orders, returns, customer\_interactions, abandoned\_carts, campaigns, shipping

1. **Revenue & GMV Reports**

-- Daily/weekly/monthly GMV

SELECT strftime('%Y-%m-%d', order\_date) AS day, SUM(total\_amount) AS daily\_gmv

FROM orders

GROUP BY day

ORDER BY day DESC;

-- Revenue by category

SELECT p.category, SUM(s.total\_price) AS revenue

FROM sales s

JOIN products p ON s.product\_id = p.product\_id

GROUP BY p.category

ORDER BY revenue DESC;

-- Returns-adjusted revenue

SELECT

strftime('%Y-%m', o.order\_date) AS month,

SUM(o.net\_amount) AS gross\_revenue,

IFNULL(SUM(r.refund\_amount), 0) AS total\_refunds,

SUM(o.net\_amount) - IFNULL(SUM(r.refund\_amount), 0) AS net\_revenue

FROM orders o

LEFT JOIN returns r ON o.order\_id = r.order\_id

GROUP BY month

ORDER BY month DESC;

1. **Sales Conversion Funnel**

-- Funnel counts from customer\_interactions + abandoned carts + orders

SELECT

COUNT(DISTINCT ci.session\_id) FILTER (WHERE ci.interaction\_type = 'Page View') AS site\_visits,

COUNT(DISTINCT ci.session\_id) FILTER (WHERE ci.interaction\_type = 'Add to Cart') AS product\_views,

COUNT(DISTINCT ac.session\_id) AS cart\_additions,

COUNT(DISTINCT o.order\_id) AS completed\_orders

FROM customer\_interactions ci

LEFT JOIN abandoned\_carts ac ON ci.session\_id = ac.session\_id

LEFT JOIN orders o ON ci.customer\_id = o.customer\_id;

-- Abandoned cart analysis

SELECT

COUNT(\*) AS total\_abandoned,

SUM(recovered) AS recovered\_carts,

ROUND(100.0 \* SUM(recovered) / COUNT(\*), 2) AS recovery\_rate

FROM abandoned\_carts;

-- Conversion by device/channel

SELECT device\_type, channel,

COUNT(DISTINCT o.order\_id) AS orders,

COUNT(DISTINCT c.customer\_id) AS unique\_customers

FROM orders o

JOIN customers c ON o.customer\_id = c.customer\_id

GROUP BY device\_type, channel;

1. **Product Sales Performance**

-- Top-selling SKUs

SELECT p.sku, p.product\_name, SUM(s.quantity) AS units\_sold, SUM(s.total\_price) AS total\_revenue

FROM sales s

JOIN products p ON s.product\_id = p.product\_id

GROUP BY p.product\_id

ORDER BY units\_sold DESC

LIMIT 10;

-- Sales by product category

SELECT p.category, COUNT(DISTINCT s.sale\_id) AS total\_sales, SUM(s.total\_price) AS revenue

FROM sales s

JOIN products p ON s.product\_id = p.product\_id

GROUP BY p.category;

-- Bundled vs single item (based on order ID having multiple products)

SELECT

CASE WHEN COUNT(s.product\_id) > 1 THEN 'Bundle' ELSE 'Single' END AS order\_type,

COUNT(DISTINCT s.order\_id) AS order\_count

FROM sales s

GROUP BY s.order\_id;

1. **Sales by Channel & Campaign**

-- Sales by source/channel

SELECT o.channel, COUNT(DISTINCT o.order\_id) AS orders, SUM(o.net\_amount) AS revenue

FROM orders o

GROUP BY o.channel

ORDER BY revenue DESC;

-- Promo/campaign performance

SELECT c.campaign\_name, c.channel, COUNT(o.order\_id) AS orders,

SUM(o.net\_amount) AS revenue, c.spent, c.revenue AS campaign\_revenue,

ROUND((c.revenue - c.spent) / NULLIF(c.spent, 0), 2) AS roi

FROM campaigns c

LEFT JOIN orders o ON c.campaign\_id = o.campaign\_id

GROUP BY c.campaign\_id

ORDER BY roi DESC;

-- Campaign ROI by segment

SELECT c.target\_segment, AVG(c.roi) AS avg\_roi

FROM campaigns c

GROUP BY c.target\_segment;

1. **Customer Acquisition & Retention Sales View**

-- New vs returning customer revenue

SELECT is\_first\_order,

COUNT(order\_id) AS total\_orders,

SUM(net\_amount) AS total\_revenue

FROM orders

GROUP BY is\_first\_order;

-- First purchase trends

SELECT strftime('%Y-%m', registration\_date) AS cohort,

COUNT(\*) FILTER (WHERE is\_active = 1) AS active\_customers,

COUNT(\*) AS total\_customers

FROM customers

GROUP BY cohort;

-- Repeat purchase rate

SELECT

COUNT(DISTINCT customer\_id) FILTER (WHERE order\_count > 1) \* 1.0 / COUNT(DISTINCT customer\_id) AS repeat\_purchase\_rate

FROM (

SELECT customer\_id, COUNT(\*) AS order\_count

FROM orders

GROUP BY customer\_id

) sub;

1. **Regional & Fulfillment-Based Sales Insights**

-- Sales by fulfillment center

SELECT fulfillment\_center, SUM(net\_amount) AS revenue, COUNT(\*) AS orders

FROM orders

GROUP BY fulfillment\_center;

-- Regional promotion performance

SELECT o.shipping\_state, c.campaign\_name, SUM(o.net\_amount) AS regional\_revenue

FROM orders o

JOIN campaigns c ON o.campaign\_id = c.campaign\_id

GROUP BY o.shipping\_state, c.campaign\_name;

-- SLA impact: delayed vs on-time delivery

SELECT

CASE

WHEN julianday(delivery\_date) > julianday(estimated\_delivery) THEN 'Delayed'

ELSE 'On-Time'

END AS sla\_status,

COUNT(\*) AS deliveries

FROM shipping

WHERE delivery\_date IS NOT NULL

GROUP BY sla\_status;

**📦** **Inventory Optimization (for eCommerce)**

Data\_sources\_required: inventory, products, sales, orders, campaigns

1. **Stock Availability & Health**

-- Real-time stock levels by SKU

SELECT p.sku, p.product\_name, i.quantity\_available, i.warehouse\_id

FROM inventory i

JOIN products p ON i.product\_id = p.product\_id

ORDER BY i.quantity\_available ASC;

-- Stockout and overstock alerts

SELECT

p.product\_name,

i.quantity\_available,

p.reorder\_point,

p.safety\_stock,

CASE

WHEN i.quantity\_available = 0 THEN 'Stockout'

WHEN i.quantity\_available < p.reorder\_point THEN 'Below Reorder Point'

WHEN i.quantity\_available > p.safety\_stock \* 2 THEN 'Overstock'

ELSE 'Normal'

END AS stock\_status

FROM inventory i

JOIN products p ON i.product\_id = p.product\_id;

-- Safety stock compliance

SELECT

p.product\_name,

i.quantity\_available,

p.safety\_stock,

CASE

WHEN i.quantity\_available < p.safety\_stock THEN 'Non-Compliant'

ELSE 'Compliant'

END AS compliance\_status

FROM inventory i

JOIN products p ON i.product\_id = p.product\_id;

1. **Demand Forecasting & Seasonality**

-- Monthly sales volume (can be input for ML forecasting)

SELECT strftime('%Y-%m', s.sale\_date) AS month, s.product\_id, p.product\_name, SUM(s.quantity) AS total\_sold

FROM sales s

JOIN products p ON s.product\_id = p.product\_id

GROUP BY month, s.product\_id

ORDER BY month DESC;

-- Seasonal trends by category

SELECT

p.category,

strftime('%m', s.sale\_date) AS month,

SUM(s.quantity) AS units\_sold

FROM sales s

JOIN products p ON s.product\_id = p.product\_id

GROUP BY p.category, month

ORDER BY p.category, month;

-- Campaign event stock projection (compare campaign period sales vs stock)

SELECT

p.product\_name,

i.quantity\_available,

SUM(s.quantity) AS campaign\_demand

FROM sales s

JOIN products p ON s.product\_id = p.product\_id

JOIN inventory i ON p.product\_id = i.product\_id

JOIN orders o ON s.order\_id = o.order\_id

JOIN campaigns c ON o.campaign\_id = c.campaign\_id

WHERE s.sale\_date BETWEEN c.start\_date AND c.end\_date

GROUP BY p.product\_id

ORDER BY campaign\_demand DESC;

1. **Inventory Turnover & Aging**

-- Inventory velocity by product

SELECT

p.product\_name,

SUM(s.quantity) AS total\_units\_sold,

AVG(i.quantity\_available) AS avg\_stock,

ROUND(SUM(s.quantity) \* 1.0 / NULLIF(AVG(i.quantity\_available), 0), 2) AS turnover\_ratio

FROM sales s

JOIN products p ON s.product\_id = p.product\_id

JOIN inventory i ON p.product\_id = i.product\_id

GROUP BY p.product\_id;

-- Aging inventory

SELECT

p.product\_name,

julianday('now') - julianday(p.launch\_date) AS age\_days,

i.quantity\_available

FROM products p

JOIN inventory i ON p.product\_id = i.product\_id

WHERE i.quantity\_available > 0

ORDER BY age\_days DESC;

-- Days on hand

SELECT

p.product\_name,

i.quantity\_available,

AVG(s.quantity) AS avg\_daily\_sales,

ROUND(i.quantity\_available / NULLIF(AVG(s.quantity), 0), 2) AS days\_on\_hand

FROM inventory i

JOIN products p ON i.product\_id = p.product\_id

JOIN sales s ON p.product\_id = s.product\_id

WHERE s.sale\_date >= date('now', '-30 days')

GROUP BY p.product\_id;

1. **Returns Impact on Inventory**

-- Restockable returns by product

SELECT

p.product\_name,

COUNT(r.return\_id) AS return\_count,

SUM(CASE WHEN r.restockable = 1 THEN 1 ELSE 0 END) AS restockable\_returns

FROM returns r

JOIN products p ON r.product\_id = p.product\_id

GROUP BY r.product\_id;

-- Refund rate by SKU

SELECT

p.sku,

COUNT(r.return\_id) AS total\_returns,

SUM(r.refund\_amount) AS total\_refunded,

ROUND(SUM(r.refund\_amount) / COUNT(r.return\_id), 2) AS avg\_refund,

p.return\_rate

FROM returns r

JOIN products p ON r.product\_id = p.product\_id

GROUP BY r.product\_id;

-- Returns effect on inventory planning

SELECT

p.product\_name,

SUM(r.refund\_amount) AS total\_refund\_loss,

COUNT(r.return\_id) FILTER (WHERE r.restockable = 0) AS unusable\_returns

FROM returns r

JOIN products p ON r.product\_id = p.product\_id

GROUP BY p.product\_id

ORDER BY total\_refund\_loss DESC;

1. **Procurement & Replenishment Optimization**

-- Reorder alerts

SELECT p.product\_name, i.quantity\_available, p.reorder\_point

FROM inventory i

JOIN products p ON i.product\_id = p.product\_id

WHERE i.quantity\_available < p.reorder\_point;

-- Vendor fulfillment performance

SELECT v.vendor\_name, COUNT(p.product\_id) AS products\_supplied,

AVG(v.performance\_score) AS avg\_score

FROM vendors v

JOIN products p ON v.vendor\_id = p.vendor\_id

GROUP BY v.vendor\_id

ORDER BY avg\_score DESC;

-- Inventory ROI

SELECT

p.product\_name,

p.cost,

AVG(s.total\_price / s.quantity) AS avg\_sales\_price,

ROUND((AVG(s.total\_price / s.quantity) - p.cost) / p.cost \* 100, 2) AS roi\_percent

FROM sales s

JOIN products p ON s.product\_id = p.product\_id

GROUP BY p.product\_id

ORDER BY roi\_percent DESC;

1. **Warehouse-Level Inventory Insights**

-- Inventory distribution

SELECT w.warehouse\_name, SUM(i.quantity\_available) AS total\_stock

FROM inventory i

JOIN warehouses w ON i.warehouse\_id = w.warehouse\_id

GROUP BY w.warehouse\_id;

-- Picking/packing efficiency (proxy via shipment volume)

SELECT w.warehouse\_name, COUNT(s.shipping\_id) AS shipments

FROM shipping s

JOIN warehouses w ON s.warehouse\_id = w.warehouse\_id

GROUP BY w.warehouse\_id

ORDER BY shipments DESC;

-- Stock transfer potential (imbalanced inventory)

SELECT

p.product\_name,

i.warehouse\_id,

i.quantity\_available

FROM inventory i

JOIN products p ON i.product\_id = p.product\_id

WHERE i.quantity\_available > (SELECT AVG(quantity\_available) FROM inventory WHERE product\_id = i.product\_id) \* 1.5

ORDER BY i.quantity\_available DESC;

**👥** **Employee Performance Analytics (for eCommerce Ops, CX, & Fulfillment)**

1. **Fulfillment Center Workforce Productivity**

-- Orders picked/packed per employee (based on shipping records)

SELECT e.employee\_id, e.first\_name || ' ' || e.last\_name AS employee\_name,

COUNT(s.shipping\_id) AS orders\_handled

FROM shipping s

JOIN warehouses w ON s.warehouse\_id = w.warehouse\_id

JOIN employees e ON w.manager\_id = e.employee\_id

GROUP BY e.employee\_id

ORDER BY orders\_handled DESC;

-- SLA adherence: shipping delay (dispatch time vs estimate)

SELECT

s.order\_id,

s.ship\_date,

s.estimated\_delivery,

julianday(s.estimated\_delivery) - julianday(s.ship\_date) AS estimated\_days,

julianday(s.delivery\_date) - julianday(s.ship\_date) AS actual\_days,

CASE

WHEN s.delivery\_date <= s.estimated\_delivery THEN 'On-Time'

ELSE 'Delayed'

END AS sla\_status

FROM shipping s

WHERE s.delivery\_date IS NOT NULL;

-- Shift-wise performance summary

SELECT e.shift, COUNT(s.shipping\_id) AS shipments

FROM shipping s

JOIN warehouses w ON s.warehouse\_id = w.warehouse\_id

JOIN employees e ON w.manager\_id = e.employee\_id

GROUP BY e.shift

ORDER BY shipments DESC;

1. **Customer Support Performance**

-- Tickets resolved per agent

SELECT cs.agent\_id, e.first\_name || ' ' || e.last\_name AS agent\_name,

COUNT(cs.ticket\_id) AS resolved\_tickets

FROM customer\_support cs

JOIN employees e ON cs.agent\_id = e.employee\_id

WHERE cs.status IN ('Resolved', 'Closed')

GROUP BY cs.agent\_id;

-- First Contact Resolution (FCR) rate

SELECT

AVG(CASE WHEN first\_contact\_resolution = 1 THEN 1.0 ELSE 0 END) \* 100 AS fcr\_rate

FROM customer\_support;

-- CSAT & NPS by agent/team

SELECT cs.agent\_id, AVG(cs.satisfaction\_score) AS avg\_csat, COUNT(cs.ticket\_id) AS tickets\_handled

FROM customer\_support cs

GROUP BY cs.agent\_id

ORDER BY avg\_csat DESC;

1. **Sales & Category Manager KPIs**

-- Revenue managed per salesperson

SELECT e.employee\_id, e.first\_name || ' ' || e.last\_name AS name,

SUM(s.total\_price) AS total\_sales

FROM sales s

JOIN employees e ON s.salesperson\_id = e.employee\_id

GROUP BY e.employee\_id

ORDER BY total\_sales DESC;

-- Campaign success metrics by manager (using campaign ROI)

SELECT e.employee\_id, e.first\_name || ' ' || e.last\_name AS manager\_name,

AVG(c.roi) AS avg\_campaign\_roi

FROM campaigns c

JOIN employees e ON e.employee\_id = c.campaign\_id % 50 + 1 -- Simulated mapping

GROUP BY e.employee\_id

ORDER BY avg\_campaign\_roi DESC;

-- SKU onboarding and pricing spread by brand manager

SELECT p.brand, COUNT(DISTINCT p.product\_id) AS products\_onboarded,

ROUND(MAX(p.price) - MIN(p.price), 2) AS price\_range

FROM products p

GROUP BY p.brand

ORDER BY products\_onboarded DESC;

1. **Attendance & Shift Adherence (Warehouse/Retail Staff)**

-- Absenteeism trends (inferred from missing shipping assignments or low shift counts)

SELECT e.shift, COUNT(\*) FILTER (WHERE e.overtime\_hours = 0) AS zero\_overtime\_count,

AVG(e.overtime\_hours) AS avg\_overtime

FROM employees e

GROUP BY e.shift;

-- Late punch-ins and break compliance proxy (not tracked directly, inferred from training + performance)

SELECT e.employee\_id, e.first\_name || ' ' || e.last\_name AS name,

e.training\_completed, e.performance\_score

FROM employees e

WHERE e.training\_completed = 0 AND e.performance\_score < 2.5;

-- Overtime vs output

SELECT e.employee\_id, e.overtime\_hours, e.performance\_score,

ROUND(e.performance\_score / NULLIF(e.overtime\_hours, 0), 2) AS performance\_per\_hour

FROM employees e

WHERE e.overtime\_hours > 0

ORDER BY performance\_per\_hour DESC;

1. **Training & Onboarding Effectiveness**

-- Ramp-up time (hire date to performance milestone)

SELECT e.employee\_id, e.hire\_date,

julianday('now') - julianday(e.hire\_date) AS days\_since\_hired,

e.performance\_score

FROM employees e

WHERE e.training\_completed = 1;

-- Post-training performance lift (compare trained vs untrained)

SELECT training\_completed,

AVG(performance\_score) AS avg\_score

FROM employees

GROUP BY training\_completed;

-- Certification/compliance completion

SELECT COUNT(\*) AS total\_employees,

SUM(training\_completed) AS certified,

ROUND(SUM(training\_completed) \* 1.0 / COUNT(\*) \* 100, 2) AS certification\_rate

FROM employees;

1. **Performance-Based Incentive Insights**

-- Incentive eligibility by performance

SELECT employee\_id, performance\_score,

CASE

WHEN performance\_score >= 4.0 THEN 'Eligible for Bonus'

WHEN performance\_score >= 3.0 THEN 'Consider for Bonus'

ELSE 'Not Eligible'

END AS incentive\_status

FROM employees;

-- Realized vs planned payout

SELECT employee\_id,

commission\_rate,

actual\_sales \* commission\_rate AS realized\_commission,

target\_sales \* commission\_rate AS planned\_commission

FROM employees

WHERE commission\_rate IS NOT NULL

ORDER BY realized\_commission DESC;

-- Incentives vs customer satisfaction

SELECT cs.agent\_id, AVG(cs.satisfaction\_score) AS avg\_csat, e.commission\_rate

FROM customer\_support cs

JOIN employees e ON cs.agent\_id = e.employee\_id

GROUP BY cs.agent\_id

ORDER BY avg\_csat DESC;

**📊** **Financial Reporting Suite (for eCommerce)**

1. **Profit & Loss Analysis**

-- P&L by product category

SELECT

p.category,

SUM(s.total\_price) AS revenue,

SUM(p.cost \* s.quantity) AS cogs,

SUM(s.total\_price) - SUM(p.cost \* s.quantity) AS gross\_profit

FROM sales s

JOIN products p ON s.product\_id = p.product\_id

GROUP BY p.category

ORDER BY gross\_profit DESC;

-- Net margin after returns, promotions, logistics

SELECT

o.order\_id,

o.net\_amount AS revenue,

IFNULL(r.refund\_amount, 0) AS returns,

o.discount\_amount + o.shipping\_amount AS expenses,

o.net\_amount - IFNULL(r.refund\_amount, 0) - (o.discount\_amount + o.shipping\_amount) AS net\_margin

FROM orders o

LEFT JOIN returns r ON o.order\_id = r.order\_id;

-- Profitability by customer segment

SELECT

c.customer\_segment,

SUM(s.total\_price) AS revenue,

SUM(p.cost \* s.quantity) AS cost,

SUM(s.total\_price - p.cost \* s.quantity) AS profit

FROM sales s

JOIN orders o ON s.order\_id = o.order\_id

JOIN customers c ON o.customer\_id = c.customer\_id

JOIN products p ON s.product\_id = p.product\_id

GROUP BY c.customer\_segment

ORDER BY profit DESC;

1. **Cash Flow & Working Capital Reports**

-- Cash flow from operations (weekly)

SELECT strftime('%Y-%W', transaction\_date) AS week,

SUM(CASE WHEN transaction\_type = 'Revenue' THEN amount ELSE 0 END) AS inflow,

SUM(CASE WHEN transaction\_type = 'Expense' THEN amount ELSE 0 END) AS outflow,

SUM(amount) AS net\_cash\_flow

FROM financial\_transactions

GROUP BY week

ORDER BY week DESC;

-- Inventory financing and payment cycle proxy

SELECT

p.product\_name,

AVG(p.cost) AS avg\_cost,

AVG(s.total\_price / s.quantity) AS avg\_sale\_price,

ROUND((AVG(s.total\_price / s.quantity) - AVG(p.cost)) \* 30, 2) AS estimated\_monthly\_margin

FROM products p

JOIN sales s ON p.product\_id = s.product\_id

GROUP BY p.product\_id;

-- Refund cash outflow

SELECT

strftime('%Y-%m', return\_date) AS month,

SUM(refund\_amount) AS total\_refunded

FROM returns

GROUP BY month

ORDER BY month DESC;

1. **Sales & Revenue Accounting**

-- Bookings vs billings vs realized revenue

SELECT

strftime('%Y-%m', o.order\_date) AS month,

COUNT(DISTINCT o.order\_id) AS bookings,

COUNT(DISTINCT p.payment\_id) FILTER (WHERE p.status = 'Completed') AS billings,

SUM(p.amount) FILTER (WHERE p.status = 'Completed') AS revenue

FROM orders o

LEFT JOIN payments p ON o.order\_id = p.order\_id

GROUP BY month;

-- Payment method distribution

SELECT payment\_method, COUNT(\*) AS total\_transactions, SUM(amount) AS total\_value

FROM payments

GROUP BY payment\_method

ORDER BY total\_value DESC;

-- Marketplace vs owned-site revenue

SELECT

channel,

SUM(net\_amount) AS revenue

FROM orders

GROUP BY channel

ORDER BY revenue DESC;

1. **Cost of Goods Sold (COGS) & Logistics Cost**

-- COGS by product line

SELECT p.category, SUM(p.cost \* s.quantity) AS total\_cogs

FROM sales s

JOIN products p ON s.product\_id = p.product\_id

GROUP BY p.category

ORDER BY total\_cogs DESC;

-- Fulfillment/shipping cost per order

SELECT o.order\_id, s.shipping\_cost

FROM shipping s

JOIN orders o ON s.order\_id = o.order\_id;

-- Packaging and delivery cost trends

SELECT strftime('%Y-%m', ship\_date) AS month,

AVG(s.shipping\_cost) AS avg\_shipping\_cost

FROM shipping s

GROUP BY month

ORDER BY month DESC;

1. **Returns & Refunds Financial Impact**

-- Value and volume of returns

SELECT strftime('%Y-%m', return\_date) AS month,

COUNT(\*) AS num\_returns,

SUM(refund\_amount) AS refund\_value

FROM returns

GROUP BY month

ORDER BY month DESC;

-- Over-refund or leakage risk

SELECT return\_id, refund\_amount, processing\_cost,

CASE

WHEN refund\_amount > 100 THEN 'High Risk'

WHEN refund\_amount > 50 THEN 'Medium Risk'

ELSE 'Low Risk'

END AS refund\_risk

FROM returns;

-- Fraudulent return detection proxy (frequent high refunds)

SELECT customer\_id, COUNT(\*) AS num\_returns,

SUM(refund\_amount) AS total\_refunded

FROM returns

GROUP BY customer\_id

HAVING COUNT(\*) > 5 AND SUM(refund\_amount) > 500;

1. **Tax & Compliance Reporting**

-- GST/VAT reports by state (assuming 8% tax on orders)

SELECT shipping\_state,

SUM(tax\_amount) AS total\_tax\_collected

FROM orders

GROUP BY shipping\_state

ORDER BY total\_tax\_collected DESC;

-- Marketplace TDS/TCS reconciliation proxy

SELECT o.channel, SUM(p.processing\_fee) AS platform\_fee

FROM orders o

JOIN payments p ON o.order\_id = p.order\_id

WHERE o.channel IN ('Marketplace')

GROUP BY o.channel;

-- E-invoice compliance (inferred from order + payment status)

SELECT o.order\_id, p.status, p.transaction\_id

FROM orders o

JOIN payments p ON o.order\_id = p.order\_id

WHERE p.status = 'Completed'

ORDER BY o.order\_date DESC

LIMIT 100;