

31 DAYS OF MYSQL

DATA

December

31 Days. 31 MySQL Projects. One Skill Level Up

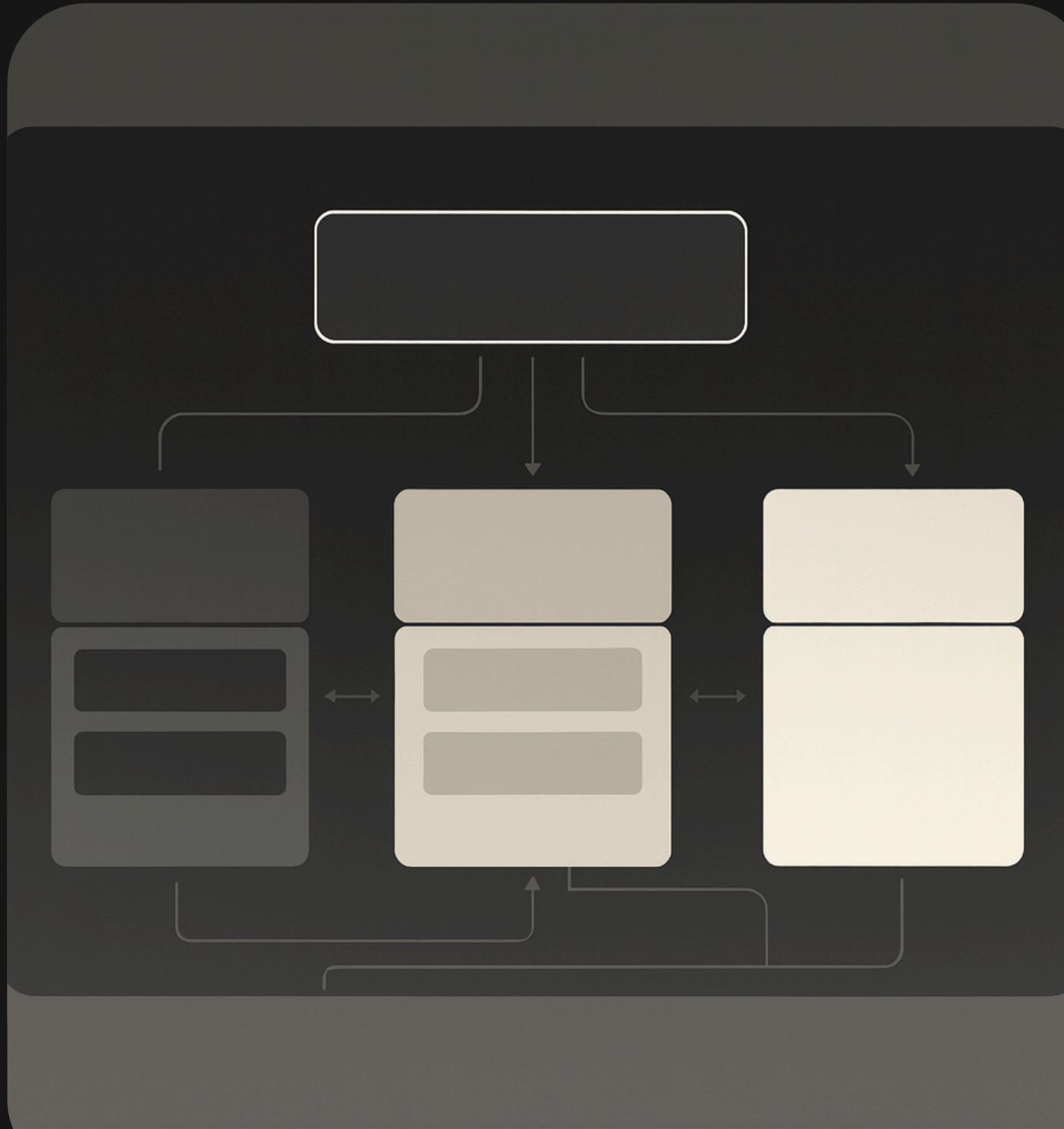
* Day 3 — Banking Dataset Project *



PROJECT OVERVIEW

This project focused on analyzing a banking-style transactional database containing customers, merchants, and financial transactions. Using SQL, the goal was to uncover patterns around spending behavior, merchant performance, fraud detection, refund patterns, and geographic risk.

DATABASE SCHEMA



transactions

- txn_id
- account_id
- merchant_id
- amount
- txn_type
- txn_date
- txn_time
- city
- is_fraud

merchants

- account_id
- name
- city
- account_open_date
- account_type
- balance

accounts

- account_id
- name
- city
- account_open_d
ate
- account_type
- balance

Query Group 1: Revenue & Monthly Trends

Q1: Top 20 Highest-Spending Customers

```
select a.account_id, a.name,  
sum(amount) as revenue  
from accounts a  
join transactions t  
on a.account_id = t.account_id  
where txn_type = 'debit'  
group by a.account_id, a.name  
order by revenue desc  
limit 20;
```

Q2 : Monthly Transaction Trend (Volume + Total Amount)

```
select month(txn_date) as month,  
count(txn_id) as txn_volume,  
sum(amount) as total_amount  
from transactions  
group by month(txn_date)  
order by month asc;
```



Query Group 2: Merchant & Fraud Detection Metrics

Query 3 : Top Merchants by Transaction Volume

```
select m.merchant_id, m.merchant_name,  
count(t.txn_id) as volume  
from merchants m  
join transactions t  
on m.merchant_id = t.merchant_id  
group by m.merchant_id,  
m.merchant_name  
order by volume desc;
```

Query 5 : Accounts with Unusually High Daily Transaction Count

```
select account_id,  
date(txn_date) as txn_day,  
count(*) as txn_count  
from transactions  
group by account_id, date(txn_date)  
having txn_count > 10  
order by txn_count desc;
```

Query 4 : Suspicious High-Value Transactions (>3x Merchant Avg)

```
with cte as (select merchant_id,  
avg(amount) as avg_amt  
from transactions  
group by merchant_id)  
  
select t.txn_id, t.merchant_id,  
t.amount as txn_amount, c.avg_amt  
from transactions t  
join cte c on t.merchant_id = c.merchant_id  
where t.amount > 3 * c.avg_amt;
```

Query Group 3: Customer Behavior & Activity

Q6: Cities with Highest Fraud Rate

```
select city,  
(count(case when is_fraud = 1 then 1 end) *  
100.0 / count(*)) as fraud_rate  
from transactions  
group by city  
order by fraud_rate desc;
```



Q7: Customers who Transacted with > 10 Unique Merchants

```
select account_id,  
count(distinct merchant_id) as  
unique_merchants  
from transactions  
group by account_id  
having unique_merchants > 10;
```



Query Group 4: Location-Based Fraud Detection

1

Q8: Transactions in a Different City than Home City

```
select a.account_id, a.city as home_city,  
t.city as transaction_city  
from accounts a  
join transactions t  
on a.account_id = t.account_id  
where a.city <> t.city;
```

2

Q9: Merchants with High Refund Rates

```
select m.merchant_id, m.merchant_name,  
(sum(case when t.txn_type = 'refund' then 1 end) * 100.0 /  
count(*)) as refund_rate  
from transactions t  
join merchants m on t.merchant_id = m.merchant_id  
group by m.merchant_id, m.merchant_name  
order by refund_rate desc;
```



Query Group 5: Refund & Risk Behavior



Query 10 : Accounts with Many Refunds but Few Purchases

```
select account_id,  
sum(case when txn_type = 'refund' then 1  
end) as refund_count,  
sum(case when txn_type = 'debit' then 1  
end) as purchase_count  
from transactions  
group by account_id  
having refund_count > 3 and  
purchase_count < 5;
```

Query 11 : Night-Time Transactions (>₹5000 between 12AM-5AM)

```
select txn_id, account_id, amount,  
txn_date, hour(txn_date) as txn_hour  
from transactions  
where hour(txn_date) between 0 and 5 and  
amount > 5000  
order by amount desc;
```

Query Group 6: Merchant Category Risk Analysis

```
select merchant_category,  
sum(amount) as total_amount,  
avg(amount) as avg_ticket_size,  
(sum(case when is_fraud = 1 then 1 end) * 100.0 / count(*)) as  
fraud_percentage  
from transactions t  
join merchants m on t.merchant_id = m.merchant_id  
group by merchant_category  
order by total_amount desc;
```





Key Skills Used

- Multi-table JOINs for connecting accounts, merchants, and transactions
- Window logic ($\text{avg} \times 3$ comparison for anomaly detection)
- Fraud analysis using conditional calculations
- Segmentation logic (high-spenders, high-refund accounts, diversified spenders)
- Geolocation mismatch detection (home city vs transaction city)
- Merchant-level risk scoring (refund %, fraud %, high-value deviations)



CONCLUSION

Day 3 provided hands-on experience with financial transaction analytics. From detecting abnormal refund patterns to identifying high-risk cities and night-time high-value transactions, the analysis reflects the type of metrics used in fraud detection, compliance, and customer risk profiling.

31 DAYS OF MYSQL, DATA DECEMBER

THANK
You!

Connect: <https://www.linkedin.com/in/altaf-khan-093730245>

GitHub: <https://github.com/Altaf0099>

* Day 3 – Banking Dataset Project *