Capstone Project - Audit

Project Code: KL-CAP-001
Client: Urban Trends Retail (Fictional)
Date: December 17 , 2025

Section A

Problem A1 - Solving the 'Ghost Data'

The client's database (fictional) is flooded with ghost data. This ghost data has been accumulated over the years due to failed transactions, cancelled transactions, accidental input, duplicated input, etc.

As the database was filled with invalid values, we have created a new table called 'mastertable' where only the valid values are entered. This was created by duplicating the values from the original table but filtering out all the *null* values or the 'ghost data' in this case.

SQL Query

```
create table mastertable as
select * from onlineretail
where customerid is not null
```

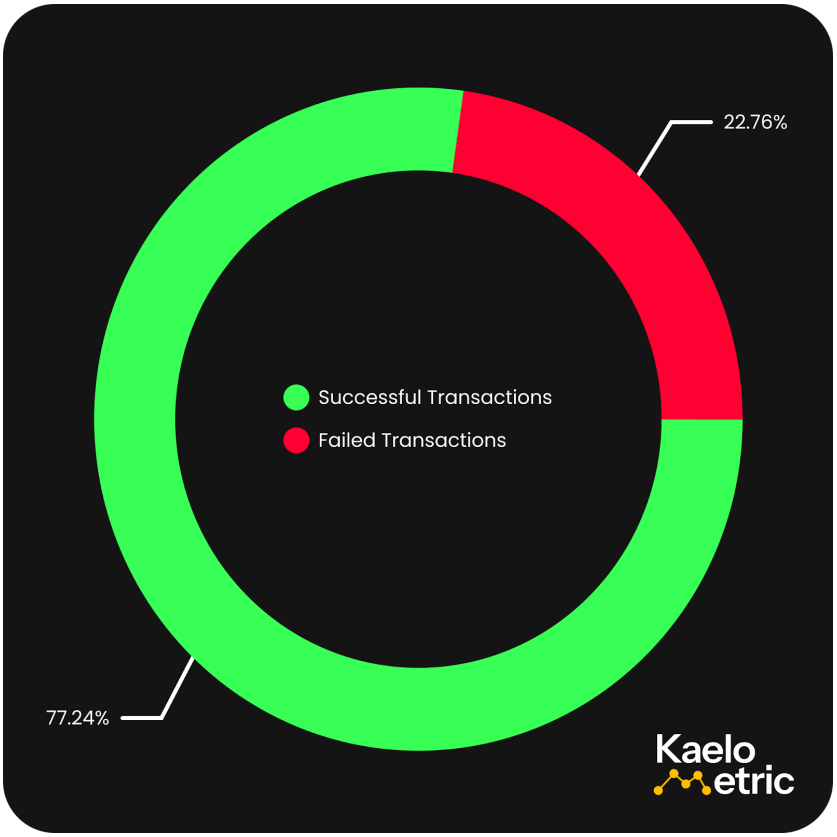
Now, to count the number of 'null' records in Customer ID was done by

SQL Query

```
select count(*) from onlineretail
where customerid is null
```

The count came out to be **243,000**
In comparison, the actual count of rows in the original database was **1,067,371**
This means, approximately **22.766%** of the total transactions either failed or were cancelled.

The following pie-chart represents the ratio of successful transactions to failed transactions.



Graph SAP1_01 : Representation of the ratio between failed transactions to successful transactions.

Problem A2 - Analyzing Cancelled Orders

The cancelled orders in the database have a letter 'C' prepended to all the invoice numbers. Hence, any data entry with invoice number starting with a 'C' was treated as a cancelled order.

To make it easier for future analysis, we created a table to only show the cancelled orders by using valid data from the *mastertable* we created earlier. This helps filter out orders that have a valid Customer ID.

SQL Query

```
create table cancelled_orders as
select * from mastertable
where invoice like 'C%'
```

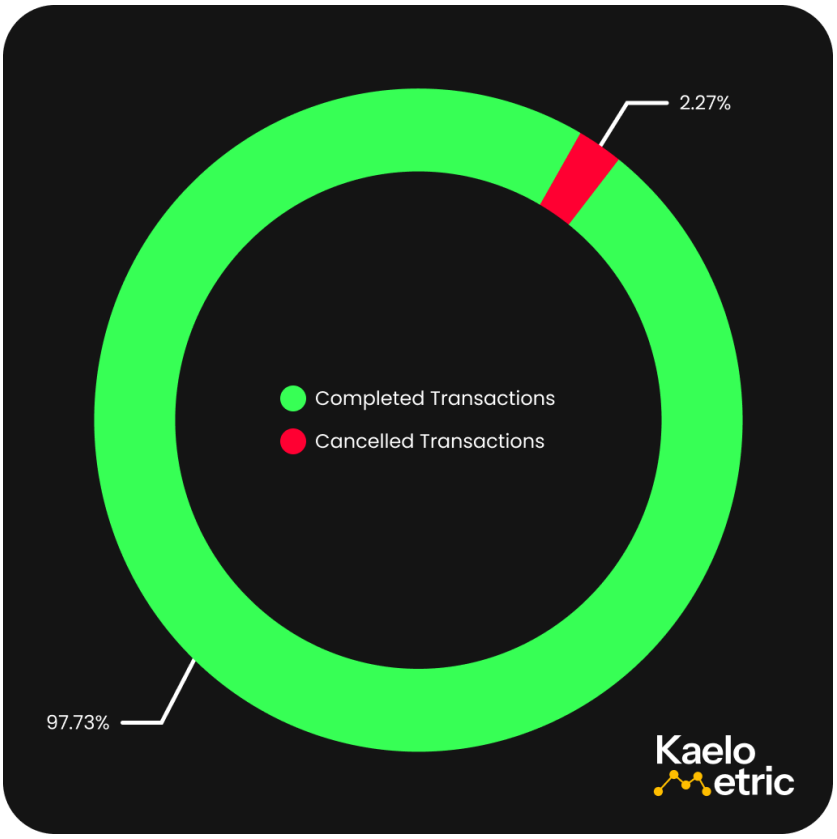
Now to compare the ratio of completed orders with cancelled orders, we first extracted the count of the cancelled transactions. The count came out to be **18,744.**

SQL Query

```
select count(*) from cancelled_orders
```

This means, 18,744 transactions out of 824,364 valid transactions were cancelled, that is approximately **2.27%** were cancelled.

To visualise this, we created the following graph which shows the ratio of the cancelled orders against the completed orders.



Graph SAP2_01: Representation of the ratio between cancelled orders vs. completed orders.

Problem A3 - Invalid Item Pricing

The database also has a set of items where the pricing is set to 0. This shows that those particular items were either given out as gifts or there was an error during the entry.

It was observed that a total of **71 transactions** had their items price set to 0. However, these items seem to have repeated.

So, to find out the distinct set of items that were given out for free, we ran the following query.

SQL Query

```
select stockcode,description,price,quantity,customerid::numeric::int
from mastertable
where price = 0
```

The above query helped us visualise the situation with items where prices were set to 0.

Here, we got to see that there were a lot of special entries where the item was a test product or a random manual entry.
The manual entries were made without any further information.

There were:

- **1,115** entries with the description set to “Manual”.
- **16** entries where the description was set to “This is a test product.”

This sums up to a total of **1,131** entries where the transaction was either of a test product or a manual entry.

Manual entries suggest the system might have failed to capture automatically and the concerned employee had to manually enter the transaction into the database.

Rolling back to the items where pricing was set to 0, we further filtered out the test products to stay out of the report so we can calculate the actual number of times an item was given out at 0 price.

SQL Query

```
select * from mastertable
where description not ilike '%test product%' and price = 0
```

This returned a total of **69 entries** where the product was not a test product but also had the price set to 0.

A total of **14,744** items were given out for free. The number was seen by running the following query.

SQL Query

```
select sum(quantity) from mastertable
where price = 0 and description not ilike '%test product%'
```

Section B

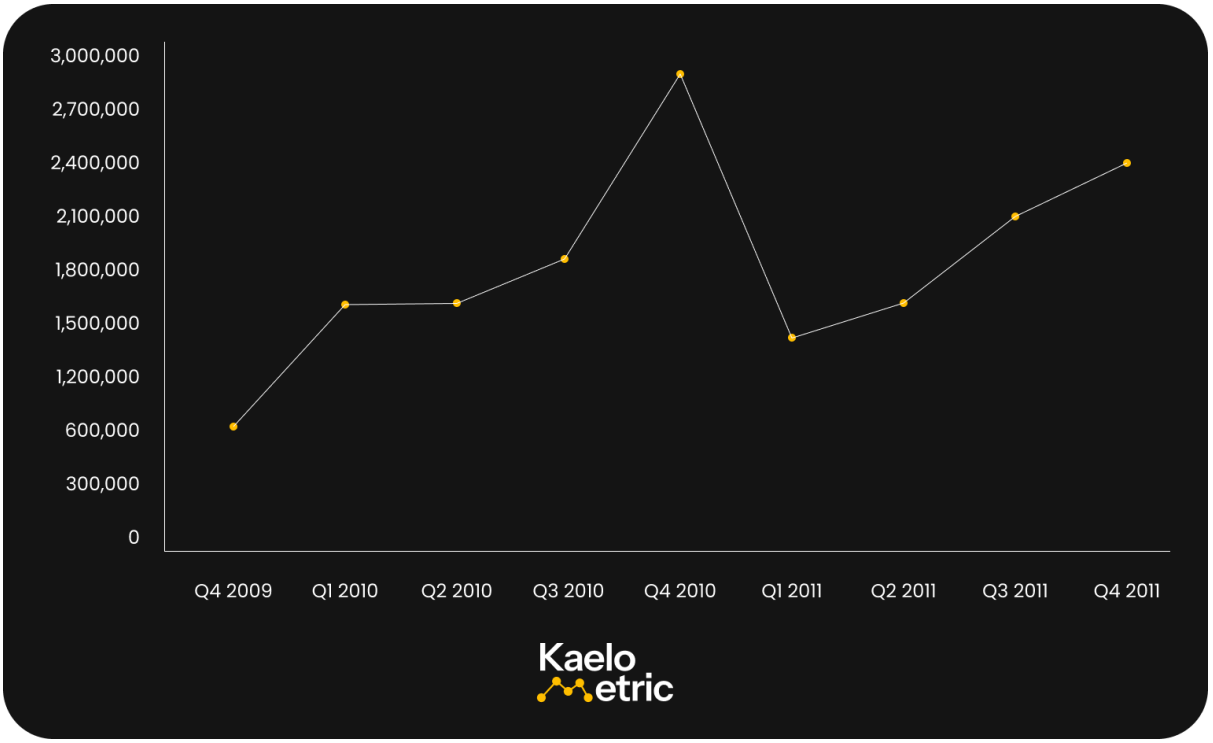
Problem B1 - Monthly Revenue Report

To retrieve a monthly revenue report of the products sold, we had to find the total sum of the product*quantity sold and group them by month.
To do that, we ran the following query where it finds the product and orders them by month in ascending order.

SQL Query

```
select round(sum(quantity*price)::numeric,2) as totalrev,
to_char(invoicedate, 'YYYY-MM') as month from mastertable
group by to_char(invoicedate, 'YYYY-MM')
order by to_char(invoicedate, 'YYYY-MM')
```

This returned 25 months of sales data from December 2009 to December 2011.
As there were a lot of months to display, we grouped it into quarters so it's easier to analyze.



Graph SBP1_01: Quarterly growth metric from 2009-2011

The Q4 of 2009 seems to be the lowest as December 2009 was the starting month in the database so the other 2 months of Q4 2009 were not considered in the calculation.

Q4 of 2010 seemed to have performed the best when compared to the rest.
So, to further analyze the months that performed the best in Q4 of 2010, we wrote the following query that shows the revenue earned in USD and the month column beside it.

SQL Query

```
select round((sum(quantity*price)::numeric),2)::money as totalrev,
to_char(invoicedate, 'MM-YYYY') as month from mastertable
where invoicedate between '2010-10-01' and '2010-12-31'
group by (to_char(invoicedate, 'MM-YYYY'))
order by (to_char(invoicedate, 'MM-YYYY'))
```

This returned 3 rows with expected outcome.

	totalrev money	month text
1	$964,989.78	10-2010
2	$1,134,879.28	11-2010
3	$859,227.37	12-2010

This shows, the month of November performed the best in Q4 of 2010 generating a total of about **$1.13 million**.

Problem B2 - Finding the top 10 best performing items

To identify the top 10 best performing items by their revenue, we ran the following query.

SQL Query

```
select description,
(round(sum(price*quantity)::numeric,2))::money as revenue,
to_char(sum(quantity),'FM9999G9999G999') as qty_sold
from mastertable

group by description
order by revenue desc
limit 10
```

This returned 10 rows as expected and the following table shows the result.

	description character varying (300)	revenue money	qty_sold text
1	REGENCY CAKESTAND 3 TIER	$269,736.70	23,446
2	WHITE HANGING HEART T-LIGHT HOLD…	$242,700.51	90,008
3	JUMBO BAG RED RETROSPOT	$134,845.16	74,564
4	ASSORTED COLOUR BIRD ORNAMENT	$126,354.18	79,434
5	POSTAGE	$112,249.10	5,078
6	PARTY BUNTING	$102,686.23	23,335
7	PAPER CHAIN KIT 50'S CHRISTMAS	$78,366.93	29,001
8	CHILLI LIGHTS	$72,229.34	15,591
9	BLACK RECORD COVER FRAME	$67,127.15	19,606
10	JUMBO BAG STRAWBERRY	$64,089.41	35,815

It was observed that **'REGENCY CAKESTAND 3 TIER'** generated a total revenue of $269,736.70 making it the highest selling item with 23,446 units sold. Note that the 'REGENCY CAKESTAND 3 TIER' however was not the highest sold in terms of quantity.

Problem B3 - Finding High-Value Customers

High-value customers are supposed to be rewarded in order to keep them coming back to the business. They are identified as loyal customers to a business if they're found coming back to the store.

The problem asks us to find the top 5 customers who have spent the highest amount of money in the year 2011.
To return their customer ID along with the total amount spent during this year, we ran the following query.

SQL Query

```
select customerid::numeric::int,
round(sum(quantity*price)::numeric,2)::money as totalspent from
mastertable
where invoicedate between '2011-01-01' and '2011-12-31'
group by customerid::numeric
order by round(sum(quantity*price)::numeric,2)::money desc
limit 5
```

This returned 5 rows as expected.

	customerid integer	totalspent money
1	14646	$270,897.14
2	18102	$228,603.88
3	17450	$185,453.33
4	14911	$125,815.49
5	12415	$123,725.45

It was identified that the customer with ID **14646** was the one who spent the highest amount of money. They spent **$270,897.14**. Following are the other 4 customers - 18102, 17450, 14911, 12415 that belong in the top 5 most-money spent list of 2011.

Problem B4 - Highest AOV excluding United Kingdom

To identify the county that generates the highest AOV excluding the UK, we first ran the following query to find the total number of orders that were placed in countries other than the United Kingdom.

SQL Query

```
select count(*) from mastertable
where country not ilike 'United Kingdom'
```

This returned a count of **83,063**.

Average order value (AOV) can be calculated by dividing the total revenue earned by the number of orders placed for each country.

SQL Query

```
select country,
round(((sum(quantity*price))/count(distinct(invoice)))::numeric,2)
as aov
from mastertable
where country not ilike 'United Kingdom'
group by country
order by aov desc
```

This returns a total of 40 countries excluding the United Kingdom where **Netherlands** had the highest AOV of **2,194** and **Saudi Arabia** with the least AOV of **65.5**.

	country character varying (150)	aov numeric
1	Netherlands	2194.10
2	Lebanon	1693.88
3	Thailand	1535.27
4	Australia	1428.45
5	Israel	1274.21
6	Denmark	1240.40
7	Lithuania	1092.29
8	RSA	966.87
9	Singapore	939.87
10	Greece	904.55
11	Switzerland	825.69
12	EIRE	795.74
13	Japan	781.72
14	Norway	740.43
15	Brazil	705.94
16	Iceland	704.16
17	Sweden	683.25
18	United Arab Emirates	647.98
19	Canada	610.38
20	Cyprus	536.95
21	West Indies	536.41
22	Channel Islands	524.73
23	Spain	488.61
24	Austria	454.46
25	France	442.42
26	Finland	434.04
27	Portugal	429.55
28	Unspecified	388.25
29	Germany	381.72
30	Belgium	347.40
31	Bahrain	338.59
32	Italy	333.47
33	Malta	324.51
34	Korea	316.61
35	Poland	277.06
36	European Community	258.35
37	USA	251.46
38	Czech Republic	141.54
39	Nigeria	70.20
40	Saudi Arabia	65.59

Section C

Problem C1 - Writing a Prompt to find the total revenue

To find the total revenue earned by the retailer, we have to write the following query which filters out the cancelled orders (Invoices with 'C' in the beginning) and prices set to 0 as they don't count towards the total revenue.

SQL Query

```
select sum(price*quantity)::numeric::money as lifetime_revenue from
mastertable
where invoice not ilike 'C%' and price > 0
```

The query above sums the product of the price and quantity, essentially returning a very long decimal number. Using the '`::money`' operator formats the returned number as a simple, international standard USD value.

Hence, the total revenue earned by the retailer excluding cancelled orders and prices of items not set to 0 is approximately **$17.7 Million**.

	lifetime_revenue money
1	$17,743,429.18

Problem C2 - Revenue Trend Chart (Monthly)

Finding the revenue vs. month was done by running the following query.

SQL Query

```
select round(sum(quantity*price)::numeric,2)::money as rev,
to_char(invoicedate, 'YYYY-MM') from mastertable
group by 2
having round(sum(quantity*price)::numeric,2) > 0
order by 2
```

Here, we summed up the product of quantity and price and grouped them by month and later ordered them by month so it starts from December of 2009 and goes on until December of 2011.

Additionally, we formatted the returned sum as *money* so the output is in the preferred USD amount format.

This returned 25 rows of 25 different months we had in the database where the revenue ranged from **$342k** to **$1.1m**.

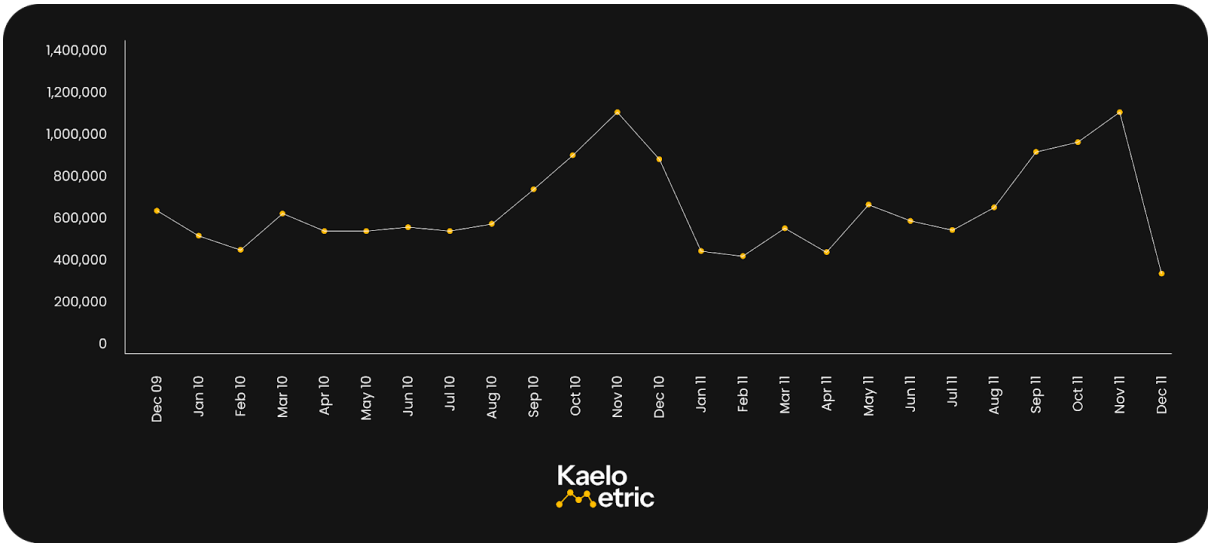
The top 3 best performing months were:

1. November 2010 ($1,134,879)
2. November 2011 ($1,132,407)
3. October 2010 ($974,603)

And the 3 worst performing months were:

1. December 2011 ($342,524)
2. April 2011 ($426,047)
3. February 2011 ($436,546)

We put this on a line graph (as requested in the problem statement) to find out the trends.



Graph SCP2_01 Revenue vs. Month Line Chart

It is observed that the sales hike up during the last 3 months of the year and fall back down during the first 3 months of the year. However, it seems like December of 2011 seemed to have severely underperformed, generating only about $342k.