Urban Collective

BI Project

Business Plan:

Introducing Urban Collective, a dynamic retail venture poised for growth and innovation in the urban fashion landscape.

Background:

Urban Collective, a prominent apparel retailer with less than 100 employees and 89 stores across Columbia, has demonstrated resilience and adaptability in the post-pandemic landscape, boasting annual sales exceeding $2 million. However, amidst plans for further expansion, there is a keen focus on understanding the intricacies of its core customer base. The marketing lead expresses concerns about the sustainability of growth, noting a reliance on hefty marketing spend to attract foot traffic. Anecdotal evidence suggests limited repeat patronage, raising questions about the alignment of customer expectations with the overall shopping experience.

BI Justification:

Urban Collective currently faces the challenge of managing data spread across various local databases, CRM systems, and spreadsheets. To streamline operations and pave the way for future expansion, a centralized repository system or data warehouse is essential. This centralized system will integrate existing data sources, accommodate increased data volume for market expansion, and continue to provide valuable business insights. The overarching objective is not only to understand the reasons behind low repeat purchases but also to comprehend the entire customer journey from initial engagement to conversion, while identifying potential markets for further expansion. To achieve this, the BI project team at Urban Collective is focused on establishing a robust database infrastructure capable of seamlessly capturing vital customer, sales, order, and product data, alongside aggregating customer feedback. This foundational framework sets the stage for the creation of a comprehensive Data Warehouse, serving as the nucleus for generating insightful BI reports. The primary goal of this BI project is to comprehend customer behavior and enhance satisfaction, leading to increased retention and repeated purchases by at least 30%. By leveraging data-driven insights, Urban Collective aims to categorize customers based on their preferences and value to the store over time, enabling informed marketing and product decisions. Addressing critical issues like the lack of returning customers and the absence of essential infrastructure is pivotal in this endeavor, underscoring the significance of implementing an operational database and data warehouse.

The implementation of a robust data warehouse infrastructure will play a pivotal role in achieving our goals. By consolidating diverse data streams from sales transactions, customer interactions, and operational metrics, the data warehouse will provide us with a comprehensive view of our business ecosystem. This holistic perspective will enable us to identify emerging trends, market dynamics, and consumer behaviors, allowing us to anticipate shifts in demand and tailor our product offerings accordingly. Moreover, the data warehouse will facilitate advanced segmentation and targeted marketing initiatives, empowering us to cater to customer demands more effectively and foster brand loyalty. Furthermore, the integration of our BI solution with the

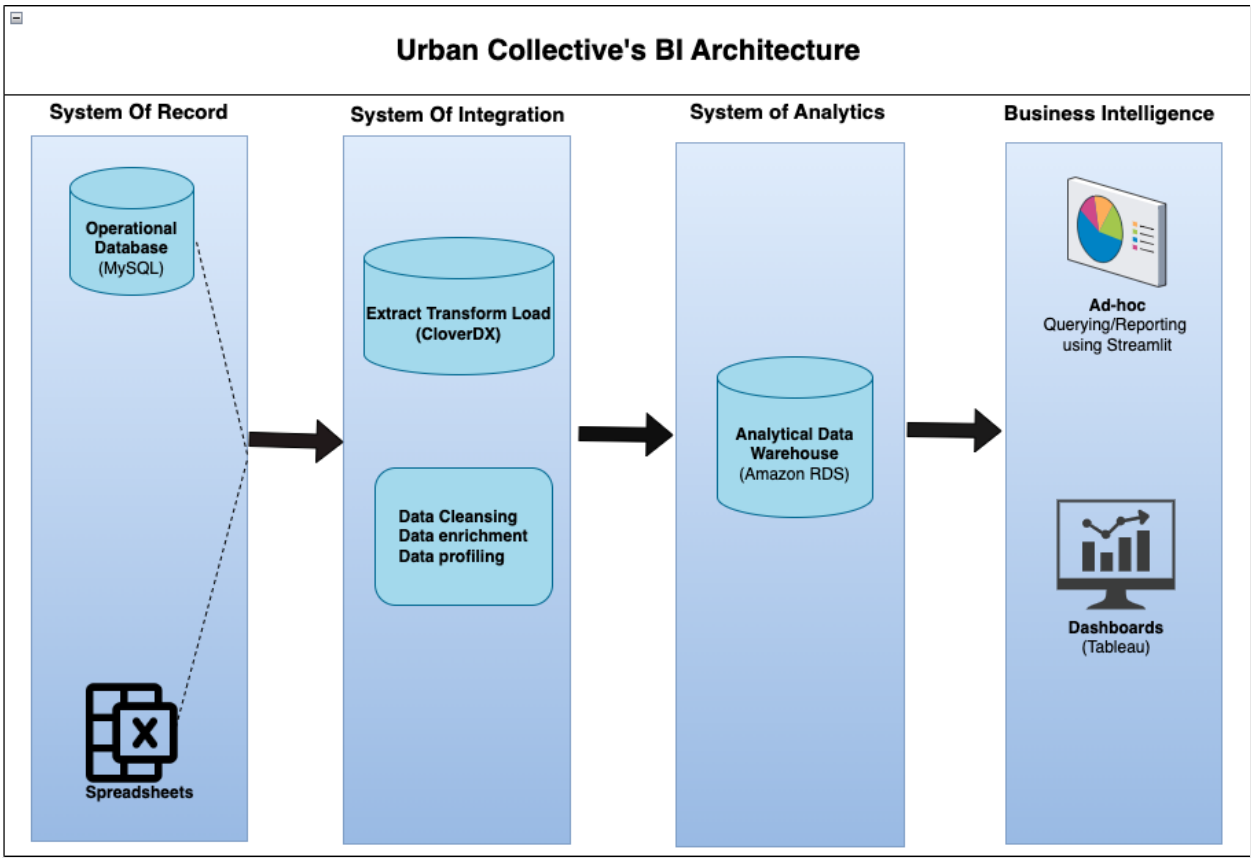
cloud will enhance accessibility, scalability, and disaster recovery, enabling us to adapt to evolving business needs and future expansion plans. By forecasting customer demands and optimizing marketing strategies, we anticipate a significant reduction in marketing costs by approximately 30% and a subsequent increase in revenue of about 20%. Additionally, the centralized repository of data will provide us with valuable insights into various aspects of our business, allowing us to explore other opportunities for growth and optimization in the future. For instance, we can analyze customer feedback to identify areas for improvement, optimize inventory management, and refine our pricing strategies.

Stakeholder consultation and user feedback are integral to refining strategies and fostering organizational growth. Engaging BI consultants and adopting an agile framework are key strategies to navigate project complexity and budget constraints. Budget constraints will necessitate the use of cost-effective tools and frameworks, leveraging existing resources wherever feasible. Documentation of a comprehensive BI model framework ensures alignment with business objectives and facilitates data-driven decision-making. In summary, our BI solution will not only help us retain our customer base and drive repeated purchases but also enable us to understand customer demand, plan for future expansion, reduce marketing costs, increase revenue, and uncover valuable business insights for continuous improvement and innovation.

BI Architecture:

Our BI architecture comprises four main components: the System of Record (Operational Database), the System of Integration (ETL Process), and the System of Analytics (Data Warehouse/Data Mart). In our System of Record, MySQL serves as the Database Management System for designing and managing the database schema. Data ingestion into the operational database occurs through various existing channels, including online orders, customer database records from CRM, manual order entries from spreadsheets, and any business processes or automated systems currently in use. This data is then integrated with an Extract Transform Load (ETL) application leveraging Integration tools, facilitating data extraction through SQL queries or database connectors. This is followed by data transformation tasks such as cleansing, aggregation, and normalization. In the System of Analytics, the transformed data can be loaded into the data warehouse using bulk loading techniques or streaming mechanisms. Online analytical processing (OLAP) or custom-built analytical applications enable querying and analytical purposes using SQL. We also have future thoughts on incorporating data marts when there are more store expansions. Additionally, we plan to utilize BI visualization tools such as Tableau and Power BI for designing visually appealing dashboards and generating detailed reports containing actionable insights. Ad hoc querying capabilities and interactive data exploration empower users to delve deeper into the data, while storytelling techniques are employed to effectively convey insights and facilitate decision-making.

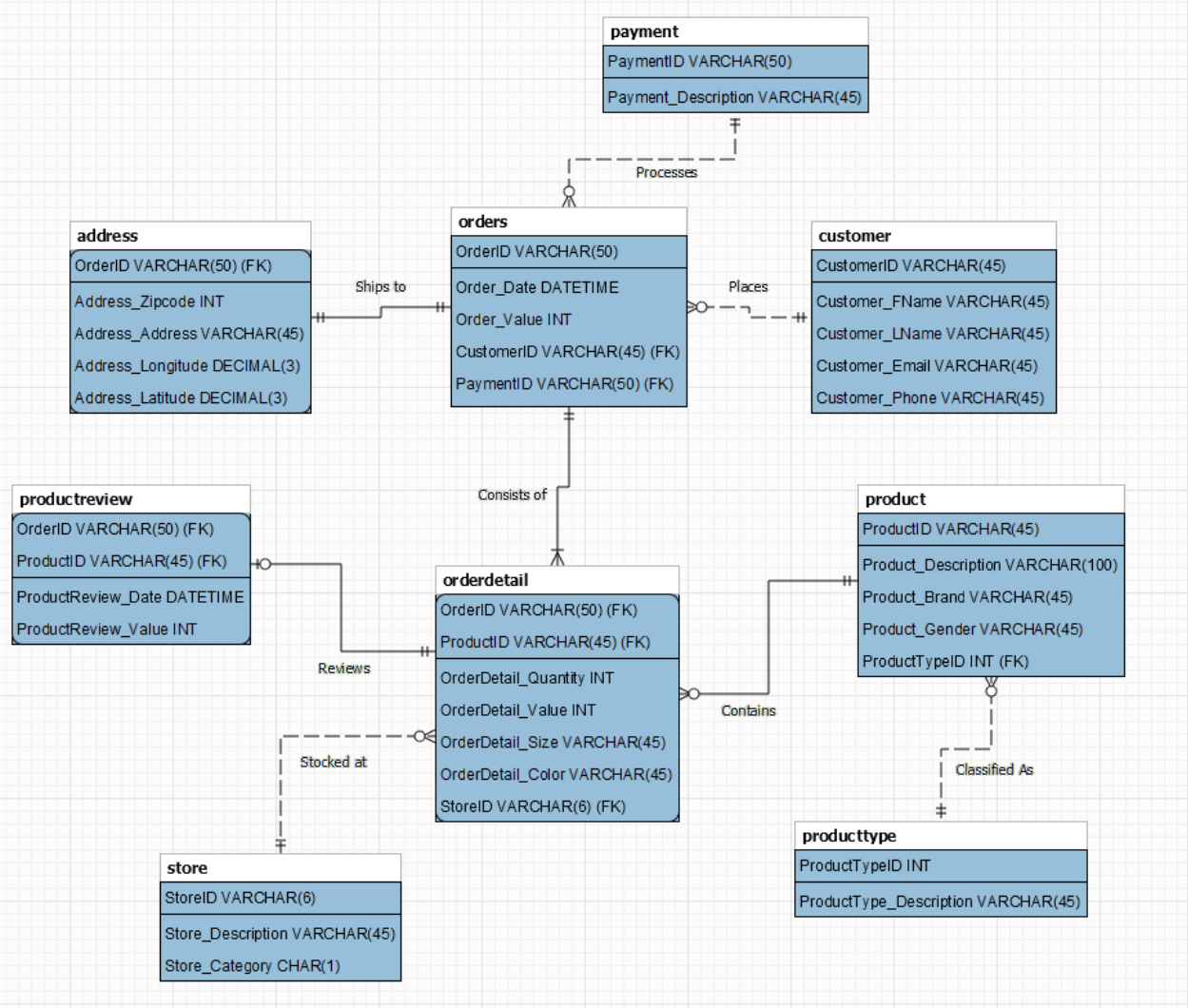
This comprehensive BI architecture ensures that Urban Collective can leverage data-driven insights to drive strategic decisions and achieve sustained growth in the competitive retail landscape. The flow chart diagram below shows the high-level implementation of the database architecture we are proposing.



ER Diagram:

When a customer visits Urban Collective's website, they browse through the available apparel and accessories. Upon finding items of interest, they proceed to place an order. This order signifies their intention to purchase specific products from Urban Collective. Each order consists of various order details, such as the quantity, size, and color of the products selected by the customer. These order details provide a comprehensive snapshot of what the customer has chosen to purchase. After placing an order, the customer provides their shipping address for delivery. Urban Collective uses this address to ensure that the ordered items are shipped to the correct location. Once the customer receives the ordered products, they may choose to leave a product review with a numeric value between one to five. These reviews provide valuable feedback to Urban Collective regarding customer satisfaction levels, product quality, and overall shopping experience.

Below is the Entity Relationship model framework that we implemented.



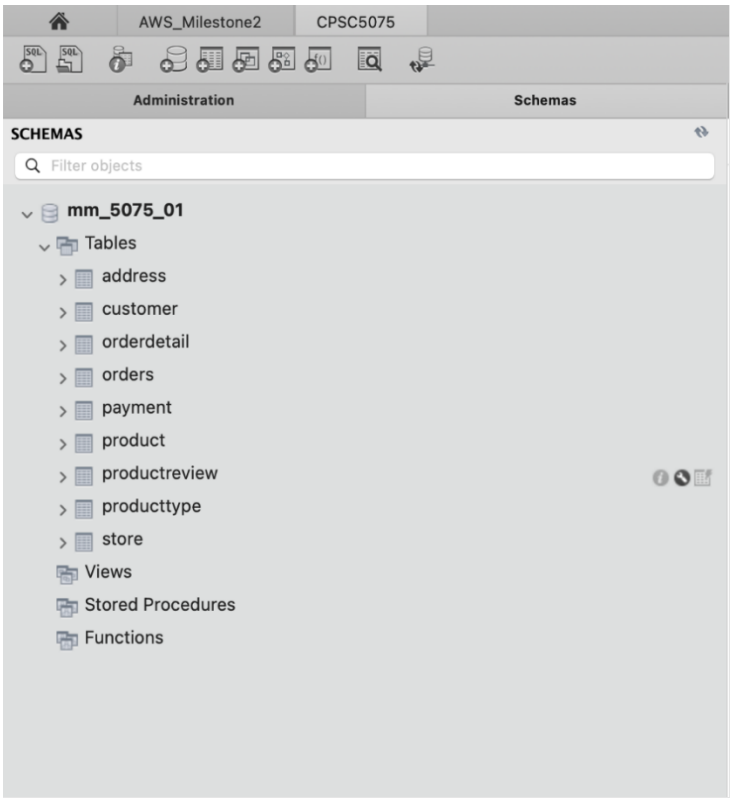
Physical Database generation and Data Population:

Source Database CSSQL:

The ER model has been forward engineered to create all the necessary tables and relationships. Data has been populated in all the tables by using csv files. The MILESTONE2DUMP.SQL file provided along with the submission zip also contains the insert statements which is in sync with mm_5075_01.

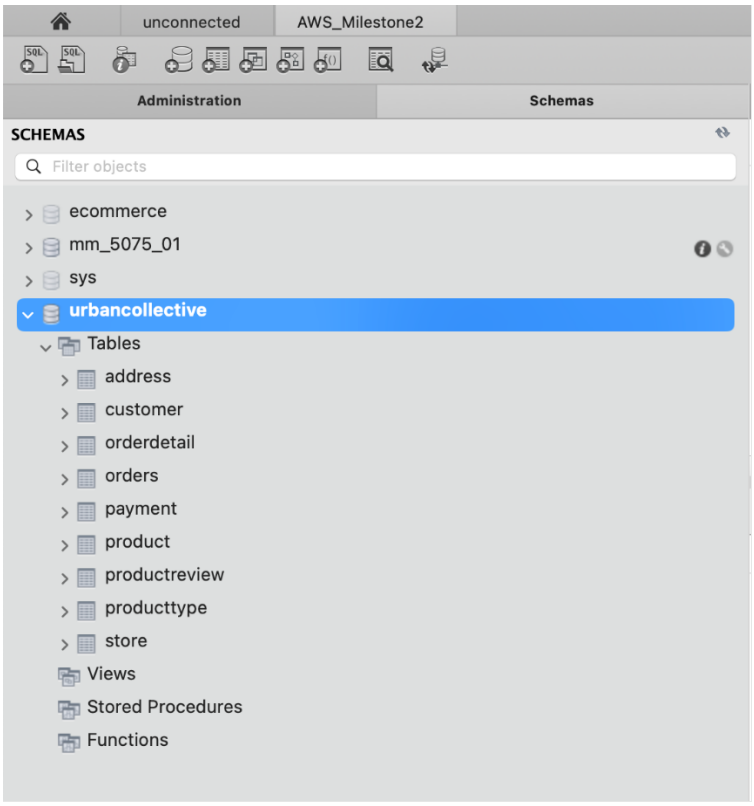
Data is first populated in the mm_5075_01 database and then cleansed and transformed and moved to cloud database urbancollective with the help of CloverDX pipeline.

Please note, this does not contain some error records which we have explicitly inserted into tables to clean using CloverDX later.



Cloud Database:

A new database called urbancollective with similar schema is created in the AWS cloud. We used our same ER model, deleted the relationships but kept foreign key columns and forward engineered to create the database urbancollective to have similar schema as our operational db mm_5075_01. Data is then populated with CloverDX and the relationships are then added in urbancollective.



Note: You might also see some more databases (please ignore them) which are just copies of our CSSQL database in Cloud as we were using that to do some Python implementation and for other experiment purposes.

Sample Data:

Below screenshots provide evidence that we have populated data in our source operational database in mm_5075_01.

```
1 • SELECT * FROM mm_5075_01.address;
```

100% 1:1

Result Grid Filter Rows: Search Edit: Export/Import:

Address_Zipcode	Address_Address	Address_Longitu...	Address_Latitu...	OrderID
63001	Cra 7#17A-00	-76	5	1400090787609-01
66001	MANZANA 22 CASA 5	-76	5	1400130787615-01
17873	Cra 10 a no 4 a - 08	-76	5	1400140787624-01
11001	Carrera 7a #135-78	-74	5	1400140787626-01
11001	Calle 134 No. 11-64 Ap. 701	-74	5	1400180787683-01
54001	Calle 11 # 1-27	-72	8	1400190787696-01
66170	Manzana 11 casa 6	-76	5	1400200787698-01
95001	Transversal 21 #13-08	-73	3	1400200787709-01

```sql
SELECT * FROM mm_5075_01.customer;
```

100% 1:1

Result Grid Filter Rows: Search Edit: Export/Import: Fetch rows

CustomerID	Customer_FName	Customer_LName	Customer_Email	Customer_Phone
1000005752	Diana Paola	Marin Osorio	diana02marin@gmail.com	5.74E+11
1000061425	Laura	Pedreros	laura.pedreross@outlook.com	+573123011456
1000121696	Sandra	Araque	sandradeicy@hotmail.com	5.73E+11
1000159085	Valeria	Lopez	valerialopez17_@hotmail.com	5.73E+11
1000160346	Juan Sebastian	Barrera	maldonado20002@gmail.com	5.73E+11
1000193246	Julian Esteban	Cadavid	jcadadavidc@unal.edu.co	5.73E+11
1000194530	Jhonatan	Espinal	crosty12345678@gmail.com	5.73E+11
1000225607	Falber Andres	Rodriguez Cardenas	falber0502@gmail.com	5.73E+11
1000351014	david	mora	dcmora10@gmail.com	5.73E+11

```sql
SELECT * FROM mm_5075_01.orderdetail;
```

100% 1:1

Result Grid Filter Rows: Search Edit: Export/Import: Fetch rows:

OrderDetail_Quant...	OrderDetail_Value	OrderDetail_Size	OrderDetail_Color	OrderID	ProductID	StoreID
1	18	M	NE	1400090787609-01	PM1101545N000	PL14P
1	151	12	T8011	1400130787615-01	Y02873P4428	PL138
1	56	34	010	1400140787624-01	M914Y000661XR05	CDEDM
1	104	32	AZC	1400140787626-01	GM2101987N000	CDEDM
1	104	M	1EI	1400140787626-01	M2013129A	ML605
1	85	S	9XX	1400180787683-01	A064890GRAM	ML417
1	20	11	BYJ2	1400180787683-01	AQYL101263	PL104
1	63	XS	BLACK	1400190787696-01	1010906	PL138
1	97	8,5	BLK	1400200787698-01	VN0A4U1KBLK	PL147

```sql
SELECT * FROM mm_5075_01.orders;
```

100% 1:1

Result Grid Filter Rows: Search Edit: Export/Import: Fetch rows:

OrderID	Order_Date	Order_Value	CustomerID	PaymentID
1400090787609-01	2024-01-01 00:00:00	18	1094967032	NEQUI
1400130787615-01	2024-01-01 00:00:00	151	10120858	Banco Caja Social
1400140787624-01	2024-01-01 00:00:00	56	75056181	Contra-entrega
1400140787626-01	2024-01-01 00:00:00	208	79948576	Banco Davivienda
1400180787683-01	2024-01-01 00:00:00	105	1020769033	Banco Colpatria
1400190787696-01	2024-01-01 00:00:00	63	1090398591	Addi
1400200787698-01	2024-01-01 00:00:00	97	18522089	Addi
1400200787709-01	2024-01-01 00:00:00	119	1000834225	SisteCredito
1400200787714-01	2024-01-01 00:00:00	14	1059904276	Contra-entrega

```
1 • SELECT * FROM mm_5075_01.payment;
```

100% 1:1

Result Grid Filter Rows: Search Edit:

PaymentID	Payment_Description
	0
Addi	Addi
Addi, Vale	Addi, Vale
American Express	American Express
Banco AV Villas	Banco AV Villas
Banco BBVA Colombia S.A.	Banco BBVA Colombia S.A.
Banco BBVA Colombia S.A., Vale	Banco BBVA Colombia S.A., Vale
Banco Caja Social	Banco Caja Social
Banco Colpatria	Banco Colpatria

```
1 • SELECT * FROM mm_5075_01.product;
```

100% 1:1

Result Grid Filter Rows: Search Edit: Export/Import:

ProductID	Product_Description	Product_Brand	Product_Gender	ProductTypeID
00C06QC859Y	LARKEE	DIESEL ADULTO	MASCULINO	201
00CLXE0852I	IAKOP	DIESEL ADULTO	MASCULINO	201
00S59V0KAQB	S-STARY	DIESEL ADULTO	MASCULINO	102
00S5PC009AX	BABHILA-HIGH-ZSP	DIESEL ADULTO	FEMENINO	201
00S6G0009IP	D-REGGY	DIESEL ADULTO	FEMENINO	201
00S6G0009RL	D-REGGY	DIESEL ADULTO	FEMENINO	201
00S6G009B11	D-REGGY	DIESEL ADULTO	FEMENINO	201
00S6G009B13	D-REGGY	DIESEL ADULTO	FEMENINO	201
00S7LY0870G	BABHILA L.32	DIESEL ADULTO	FEMENINO	201

```
1 • SELECT * FROM mm_5075_01.productreview;
```

100% 1:1

Result Grid Filter Rows: Search Edit: Export/Import:

ProductReview_Date	ProductReview_Value	OrderID	ProductID
2024-01-17 00:00:00	3	1400090787609-01	PM1101545N000
2024-01-17 00:00:00	3	1400130787615-01	Y02873P4428
2024-01-17 00:00:00	5	1400140787624-01	M914Y000661XR05
2024-01-17 00:00:00	3	1400140787626-01	GM2101987N000
2024-01-17 00:00:00	3	1400140787626-01	M2013129A
2024-01-17 00:00:00	5	1400180787683-01	A064890GRAM
2024-01-17 00:00:00	5	1400180787683-01	AQYL101263
2024-01-17 00:00:00	3	1400190787696-01	1010906
2024-01-17 00:00:00	3	1400200787698-01	VN0A4U1KBLK

```
1 • SELECT * FROM mm_5075_01.producttype;
```

100% 1:1

Result Grid Filter Rows: Search Edit: Export/Import:

ProductTypeID	ProductType_Descript...
101	CAMISETA
102	CAMISA
104	BUZO
105	CHAQUETA
107	POLO
108	TOP
201	JEAN
202	PANTALON
203	BERMUDA

```
1 • SELECT * FROM mm_5075_01.store;
```

100% 1:1

Result Grid Filter Rows: Search Edit: Export/Import:

StoreID	Store_Description	Store_Category
CDEDM	CENTRO DE DISTRIBUCION EDM	A
CDONL	TIENDAS ONLINE	A
ML210	NEW BALANCE EL TESORO MEDELLIN	A
ML211	NEW BALANCE MAYORCA MEDELLIN	A
ML212	NEW BALANCE CARACOLI BUCARAMANGA	B
ML213	NEW BALANCE DE MODA PRIME OUTLET L...	B
ML214	NEW BALANCE BUENAVISTA BARRANQUILLA	C
ML218	NEW BALANCE ARBOLEDA	A
ML219	NEW BALANCE VIVA ENVIGADO	A

RDS Connection details:

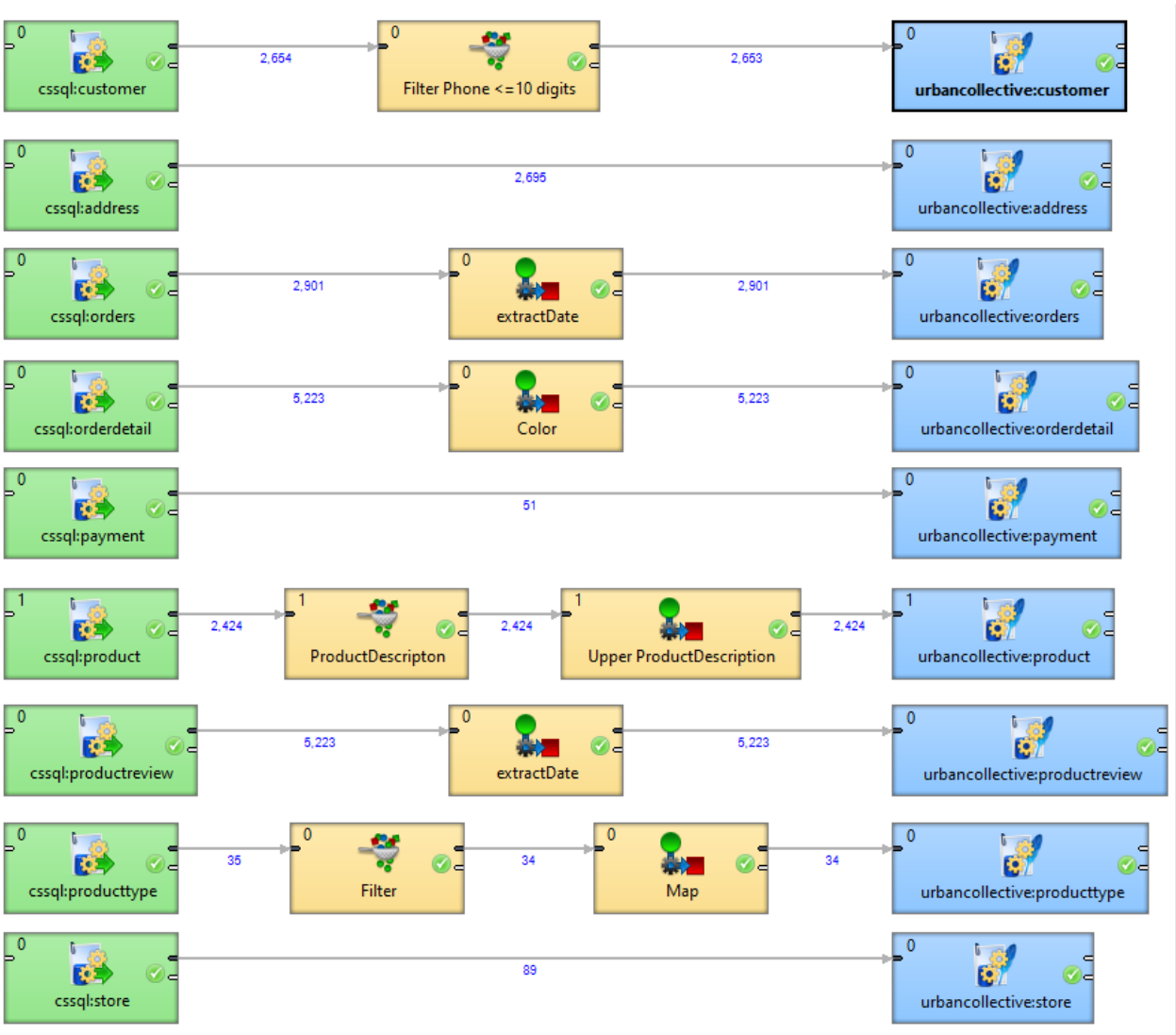
cpsc5075.c6glisumrpc8.us-east-1.rds.amazonaws.com

admin; password*

database: urbancollective

Clover DX Pipeline:

Picture of Clover DX Pipeline



Description of moving data, Cleaning, transformation:

CloverDX moves data from the Operational Database (mm_5075_01) to Analytical database (UrbanCollective) in the AWS cloud. During this migration, the source data in each Table is cleansed by filtering for invalid values (such as ‘N/A’) and mapped to standards, before being loaded into destination tables.

We have explicitly inserted some error records in **mm_5075_01** to see that CloverDX pipeline is working:

SELECT * FROM mm_5075_01.customer where CustomerID = '9999999999';

SELECT * FROM mm_5075_01.producttype where ProductTypeID = 422 and

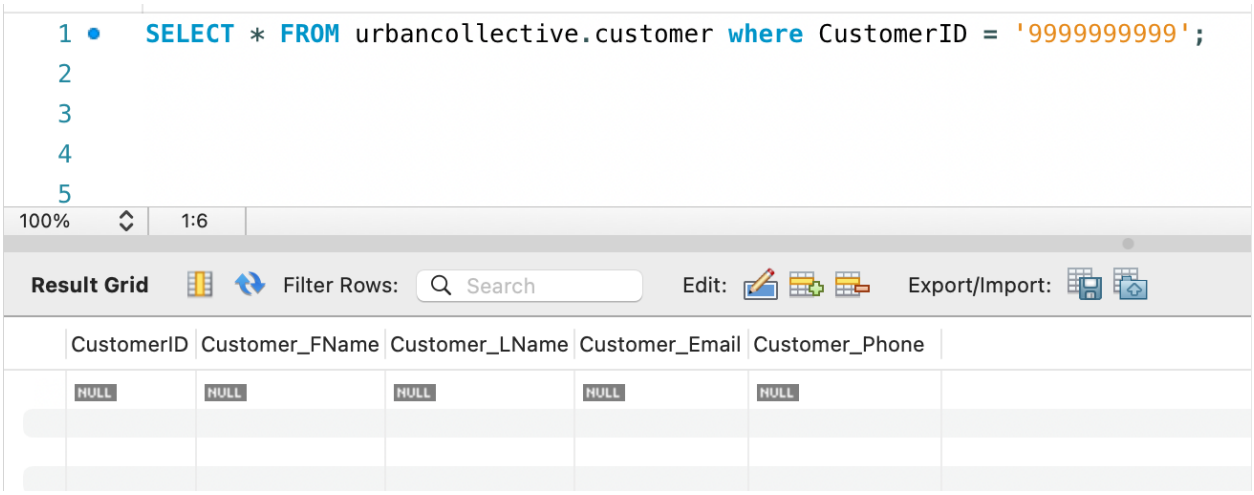
ProductType_Description ='buzo';

SELECT * FROM mm_5075_01.producttype where ProductType_Description = 'N/A';

After transformation in **urbancollective** these records are transformed:

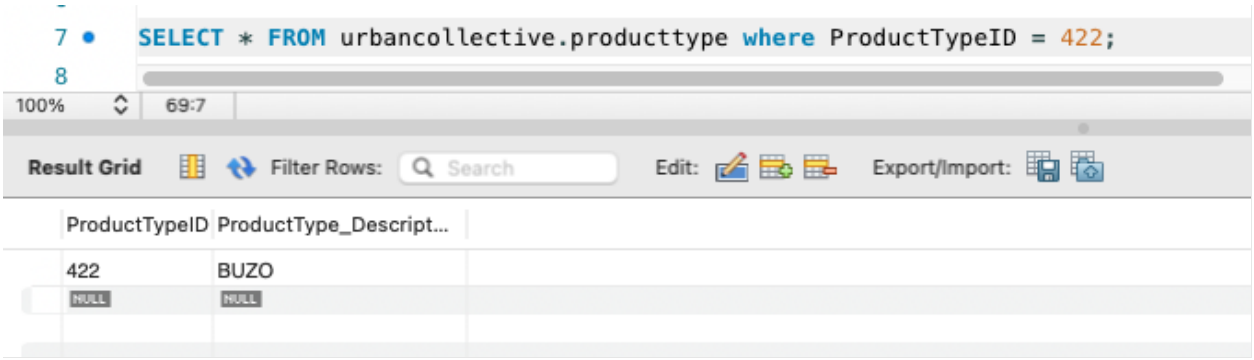
1. We remove the incorrect CustomerID record.

SELECT * FROM urbancollective.customer where CustomerID = '9999999999';



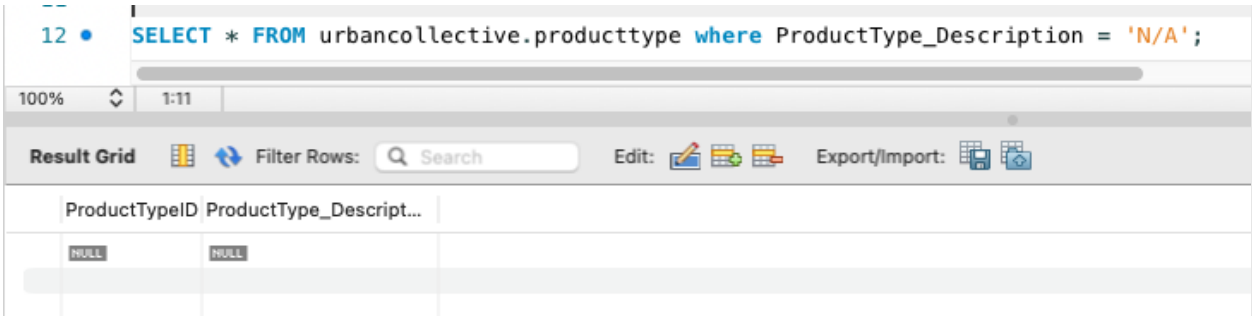
2. We capitalize the product descriprion as ‘BUZO’ to be compatible with other records.

SELECT * FROM urbancollective.producttype where ProductTypeID = 422;



3. We remove this record as it has invalid productype description as N/A.

SELECT * FROM urbancollective.producttype where ProductType_Description = 'N/A';



Data Processing :

The nine Tables of CSSQL are loaded into UrbanCollective database one at a time, with the transformations to i) filtering invalid data, ii) mapping to standardize data format and iii) mapping to standardize date:

a) Cssql:customer Table is moved into urbancollective:customer Table after removing any rows with phone number that is greater than the standard 10 digits.
b) Cssql:address Table is moved into urbancollective:address Table, as is
c) Cssql:orders Table is moved into urbancollective:orders Table after removing the time part of the Date field (because the time was found to be inaccurate).
d) Cssql:orderdetails Table is moved into urbancollective:orderdetails Table after standardizing the Color field by converting Color column to uppercase
e) Cssql:payment Table is moved into urbancollective:payment Table, as is.
f) Cssql:product Table is moved into urbancollective:product Table after filtering out invalid product descriptions and standardizing the product description field by converting product description column to uppercase
g) Cssql:productreview Table is moved into urbancollective:productreview Table, after removing the time part of the Date field (because the time was found to be inaccurate)
h) Cssql:producttypeTable is moved into urbancollective:producttype Table, after filtering out invalid product descriptions and standardizing the product description field by converting product description column to uppercase

i) Cssql:store Table is moved into urbancollective:store Table, as is

How to run the CloverDX Pipeline?

i. To run the CloverDX pipeline for urbancollective database, unzip the file ecomm2.zip. Then load the CLoverDX project into the CLoverDX Designer and open file cloverDX.grf.
ii. Confirm the connections to the source database cssql (mm_5075_01 and mm_5075_01Pass-) and destination RDS database urbancollective. [Optional: TRUNCATE all 9 tables]
iii. Ensure all jobs are enabled (right click on job and select Enable to enable). Right click on cloverDX and select Run CloverDX Graph. It takes about 15 minutes (depending on your machine resources). Confirm by comparing the resulting view of cloverDX.grf with the screenshot on previous pages.
iv. Now connect MYSQL workbench to RDS database urbancollective and confirm all the updated data is available

Analytical Queries:

Below are the analytical queries that we will be using to understand how our business is doing in terms of top orders, top products, customer reviews, store reviews etc.

1. This query retrieves products with above-average ratings based on customer reviews and have more than 5 reviews. It calculates the average product review value for each product along with the total number of reviews. The products are sorted in descending order of their average review ratings.

   By identifying products with above-average ratings and a significant number of reviews, the business can focus on promoting products that are not only highly rated but also have received enough feedback to establish credibility and reliability. This helps in building trust among potential customers and encouraging more purchases.

```
SELECT
  p.ProductID,
  p.Product_Description,
  AVG(pr.ProductReview_Value) AS Avg_ProductReview_Value,
  COUNT(pr.ProductReview_Value) AS Total_Reviews
FROM
  product p
```

```
LEFT JOIN
   productreview pr ON p.ProductID = pr.ProductID
GROUP BY
    p.ProductID, p.Product_Description
HAVING
   AVG(pr.ProductReview_Value) > (SELECT AVG(ProductReview_Value) FROM
productreview) and
   COUNT(pr.ProductReview_Value)>5
 ORDER BY
   Avg_ProductReview_Value DESC;
```

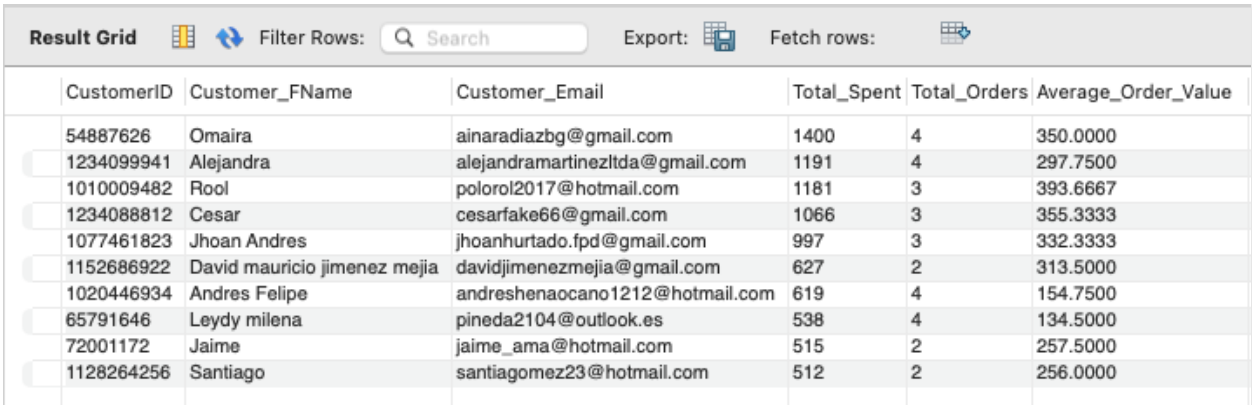
ProductID	Product_Description	Avg_ProductReview_Value	Total_Reviews
MT23220-MIB	Accelerate Singlet	4.0000	6
PS327US	Pre School 327	4.0000	6
W2011695A	VINTAGE DOWNTOWN SCRIPTED HOOD	4.0000	6
GF2400072N000	SHORT	4.0000	6
GM3100223N000	SIMPLICITY - BUZO CERRADO	4.0000	6
PF1100520N000	CAMISETA M/L	4.0000	6
WL574DT2	ZAPATILLA DE MUJER	4.0000	6
WS23230-NGO	Womens Running Accelarate 5 inch	3.8889	9

2. This query calculates the Customer Lifetime Value (CLV) for the top 10 highest-spending customers who have made at least 2 orders. It computes the total amount spent, total number of orders, and average order value for each customer.

   CLV analysis helps in identifying the most valuable customers who contribute the most revenue to the business over their lifetime. By focusing on retaining and engaging high CLV customers, the business can improve customer satisfaction, loyalty, and long-term profitability.

```
SELECT
   c.CustomerID,
   c.Customer_FName,
   c.Customer_Email,
   SUM(od.OrderDetail_Value) AS Total_Spent,
```

```
    COUNT(DISTINCT o.OrderID) AS Total_Orders,
    SUM(od.OrderDetail_Value) / COUNT(DISTINCT o.OrderID) AS Average_Order_Value
FROM
    customer c
JOIN
    orders o ON c.CustomerID = o.CustomerID
JOIN
    orderdetail od ON o.OrderID = od.OrderID
GROUP BY
    c.CustomerID
HAVING
    Total_Orders >= 2
ORDER BY
    Total_Spent DESC
LIMIT
    10;
```



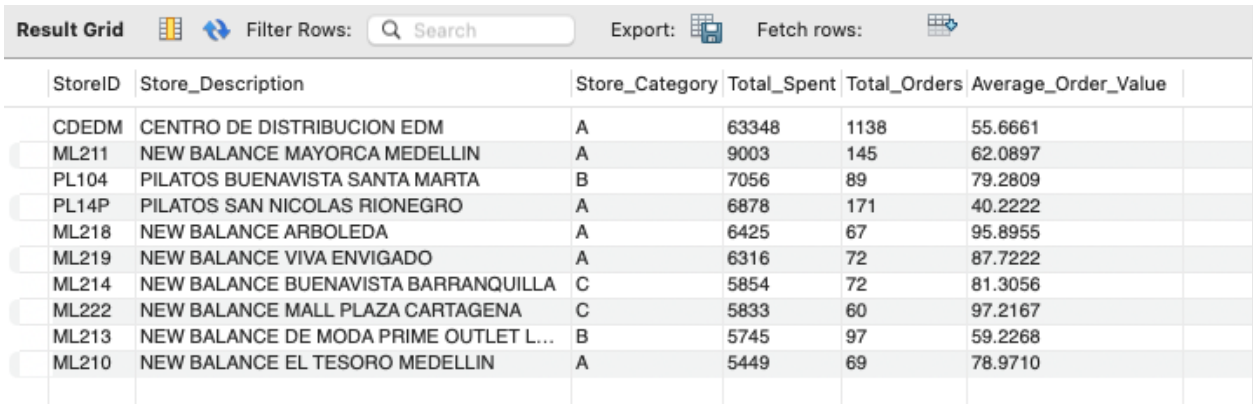
Result Grid | Filter Rows: Search | Export: | Fetch rows:

CustomerID	Customer_FName	Customer_Email	Total_Spent	Total_Orders	Average_Order_Value
54887626	Omaira	ainaradiazbg@gmail.com	1400	4	350.0000
1234099941	Alejandra	alejandramartinezltda@gmail.com	1191	4	297.7500
1010009482	Rool	polorol2017@hotmail.com	1181	3	393.6667
1234088812	Cesar	cesarfake66@gmail.com	1066	3	355.3333
1077461823	Jhoan Andres	jhoanhurtado.fpd@gmail.com	997	3	332.3333
1152686922	David mauricio jimenez mejia	davidjimenezmejia@gmail.com	627	2	313.5000
1020446934	Andres Felipe	andreshenaocano1212@hotmail.com	619	4	154.7500
65791646	Leydy milena	pineda2104@outlook.es	538	4	134.5000
72001172	Jaime	jaime_ama@hotmail.com	515	2	257.5000
1128264256	Santiago	santiagomez23@hotmail.com	512	2	256.0000

3. This query identifies the top performing Stores where the Order value higher than or equal to 50.

   This query is relevant because it identifies the top-performing stores based on total order value, which provides insights into which stores are driving the most revenue. This information is crucial for strategic decision-making, resource allocation, and identifying successful practices that can be replicated across other stores to boost overall performance.

```
SELECT
   s.StoreID,
   s.Store_Description,
   s.Store_Category,
   SUM(od.OrderDetail_Value) AS Total_Spent,
   COUNT(DISTINCT o.OrderID) AS Total_Orders,
   SUM(od.OrderDetail_Value) / COUNT(DISTINCT o.OrderID) AS Average_Order_Value
FROM
   store s
JOIN
    orderdetail od ON od.StoreID = s.StoreID
JOIN
orders o  ON o.OrderID = od.OrderID
GROUP BY
   s.StoreID
HAVING
   Total_Orders >= 50
ORDER BY
   Total_Spent DESC
LIMIT
   10;
```



StoreID	Store_Description	Store_Category	Total_Spent	Total_Orders	Average_Order_Value
CDEDM	CENTRO DE DISTRIBUCION EDM	A	63348	1138	55.6661
ML211	NEW BALANCE MAYORCA MEDELLIN	A	9003	145	62.0897
PL104	PILATOS BUENAVISTA SANTA MARTA	B	7056	89	79.2809
PL14P	PILATOS SAN NICOLAS RIONEGRO	A	6878	171	40.2222
ML218	NEW BALANCE ARBOLEDA	A	6425	67	95.8955
ML219	NEW BALANCE VIVA ENVIGADO	A	6316	72	87.7222
ML214	NEW BALANCE BUENAVISTA BARRANQUILLA	C	5854	72	81.3056
ML222	NEW BALANCE MALL PLAZA CARTAGENA	C	5833	60	97.2167
ML213	NEW BALANCE DE MODA PRIME OUTLET L...	B	5745	97	59.2268
ML210	NEW BALANCE EL TESORO MEDELLIN	A	5449	69	78.9710

4. This query identifies the top payment ID/types where the average value spent is higher than or equal to 75.

   This query is relevant because it identifies the top payment methods by total spending, which helps in understanding customer preferences for payment types. By analyzing which payment methods are associated with higher spending, businesses can optimize payment processing options, improve customer satisfaction, and potentially negotiate better terms with payment providers to enhance profitability.

```
SELECT
  p.PaymentID,
  p.Payment_Description,
  SUM(od.OrderDetail_Value) AS Total_Spent,
  COUNT(DISTINCT o.OrderID) AS Total_Orders,
  SUM(od.OrderDetail_Value) / COUNT(DISTINCT o.OrderID) AS
Average_Order_Value
FROM
  payment p
JOIN
   orders o ON o.PaymentID = p.PaymentID
JOIN
orderdetail od ON od.OrderID = o.OrderID
GROUP BY
  p.PaymentID
HAVING
  Average_Order_Value >= 75
ORDER BY
  Total_Spent DESC
 LIMIT
   10;
```

Result Grid | Filter Rows: Search | Export: | Fetch rows:

PaymentID	Payment_Description	Total_Spent	Total_Orders	Average_Order_Value
Addi	Addi	63827	700	91.1814
Mastercard	Mastercard	34881	326	106.9969
Visa	Visa	34157	312	109.4776
Bancolombia	Bancolombia	18229	222	82.1126
WompiCo	WompiCo	17362	141	123.1348
SisteCredito	SisteCredito	16596	174	95.3793
PSE	PSE	10827	112	96.6696
Banco Davivienda	Banco Davivienda	5776	70	82.5143
American Express	American Express	3111	30	103.7000
Vale	Vale	2276	27	84.2963

5. This query measures the percentage of customers who have not made more than one purchase in the last three months.

   This query helps identify customers who were active earlier but have shown inactivity in the last two month, providing insights into short-term churn. By understanding this, businesses can implement targeted retention strategies to

   re-engage these customers and reduce churn.

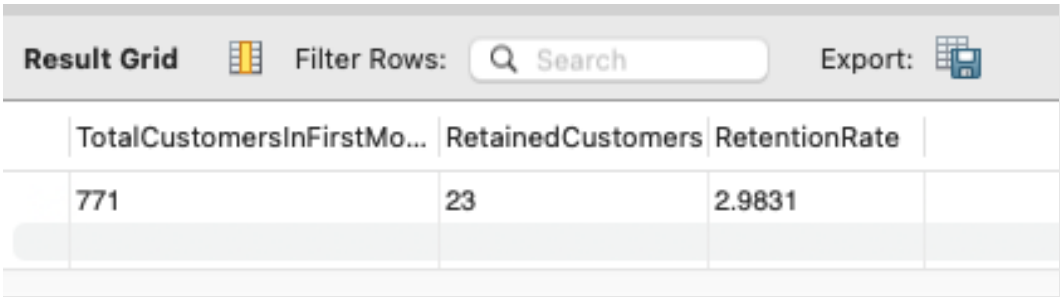
```
WITH first_month AS (
  SELECT DISTINCT CustomerID
  FROM orders
  WHERE Order_Date BETWEEN DATE_SUB(CURDATE(), INTERVAL 3 MONTH) AND
DATE_SUB(CURDATE(), INTERVAL 2 MONTH)
),

-- Find customers who made purchases in the second and third months
following_two_months AS (
  SELECT DISTINCT CustomerID
  FROM orders
  WHERE Order_Date BETWEEN DATE_SUB(CURDATE(), INTERVAL 2 MONTH) AND
CURDATE()
)

-- Calculate the retention rate
```

```
SELECT
  (SELECT COUNT(*) FROM first_month) AS TotalCustomersInFirstMonth,
  (SELECT COUNT(*) FROM first_month WHERE CustomerID IN (SELECT CustomerID
FROM following_two_months)) AS RetainedCustomers,

  (SELECT COUNT(*) FROM first_month WHERE CustomerID IN (SELECT CustomerID
FROM following_two_months)) / (SELECT COUNT(*) FROM first_month) * 100 AS
RetentionRate;
```



Note: To make this query results more viable we can restrict the dates as this data has been inserted one time and it's mock data the retention rate will keep on decreasing if we use current date. So we can analyze the first quarter of the year instead.

```
WITH first_month AS (
  SELECT DISTINCT CustomerID
  FROM orders
  WHERE Order_Date BETWEEN '2024-01-01' AND '2024-01-31'
),
following_two_months AS (
  SELECT DISTINCT CustomerID
  FROM orders
  WHERE Order_Date BETWEEN '2024-02-01' AND '2024-03-31'
)
SELECT
```

(SELECT COUNT(*) FROM first_month) AS TotalCustomersInFirstMonth,

(SELECT COUNT(*) FROM first_month WHERE CustomerID IN (SELECT CustomerID FROM following_two_months)) AS RetainedCustomers,

(SELECT COUNT(*) FROM first_month WHERE CustomerID IN (SELECT CustomerID FROM following_two_months)) / (SELECT COUNT(*) FROM first_month) * 100 AS RetentionRate;

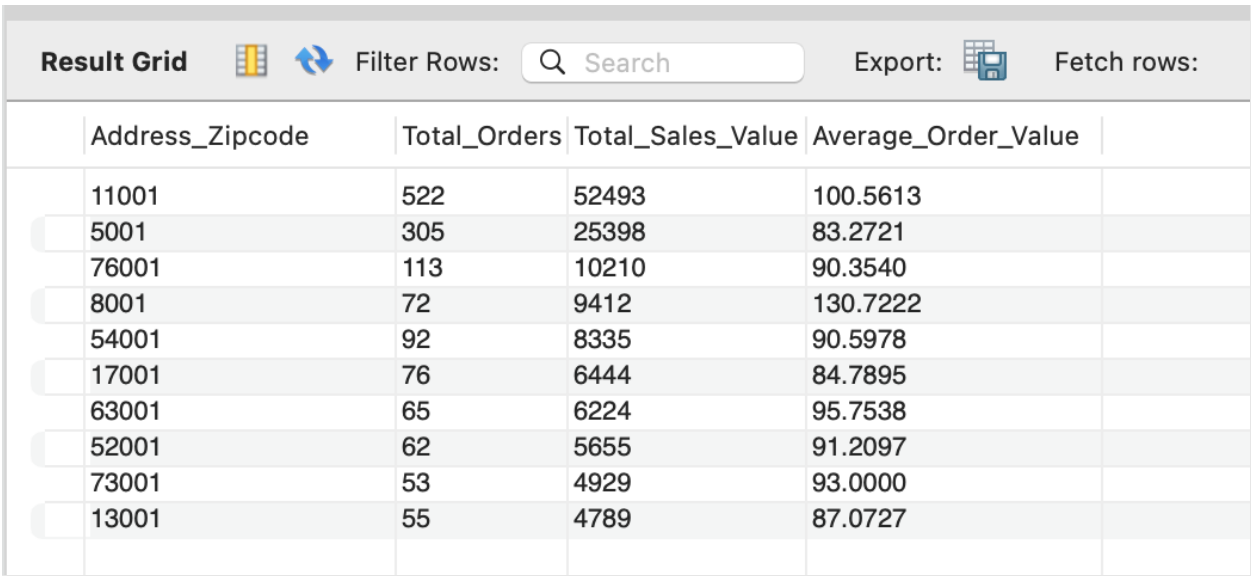
TotalCustomersInFirstMo...	RetainedCustomers	RetentionRate
1460	88	6.0274

6. This query gives top 10 Zipcode by Total Sales Value and Number of Orders.

   It helps identify regions with the highest sales and order volumes, providing insights into geographical trends in customer behavior. By understanding which regions generate the most revenue, businesses can: Tailor marketing and promotional efforts to high-performing regions. Allocate resources more effectively to boost sales in underperforming areas. Analyze regional preferences and adapt product offerings to meet local demand.

```
SELECT
  a.Address_Zipcode,
  COUNT(DISTINCT o.OrderID) AS Total_Orders,
  SUM(o.Order_Value) AS Total_Sales_Value,
  AVG(o.Order_Value) AS Average_Order_Value
FROM
  address a
JOIN
  orders o ON a.OrderID = o.OrderID
GROUP BY
  a.Address_Zipcode
ORDER BY
  Total_Sales_Value DESC
```

```
LIMIT
    10;
```



Result Grid | Filter Rows: Search | Export: | Fetch rows:

Address_Zipcode	Total_Orders	Total_Sales_Value	Average_Order_Value
11001	522	52493	100.5613
5001	305	25398	83.2721
76001	113	10210	90.3540
8001	72	9412	130.7222
54001	92	8335	90.5978
17001	76	6444	84.7895
63001	65	6224	95.7538
52001	62	5655	91.2097
73001	53	4929	93.0000
13001	55	4789	87.0727

Python Program to interact with our Database

Our current data infrastructure revolves around a single database, the Urbancollective Analytical Database, which is the result of our ETL (Extract, Transform, Load) processes performed in CloverDX. This database, hosted on Amazon Web Services (AWS), contains the processed and refined data that we use for all our analytical and reporting needs.

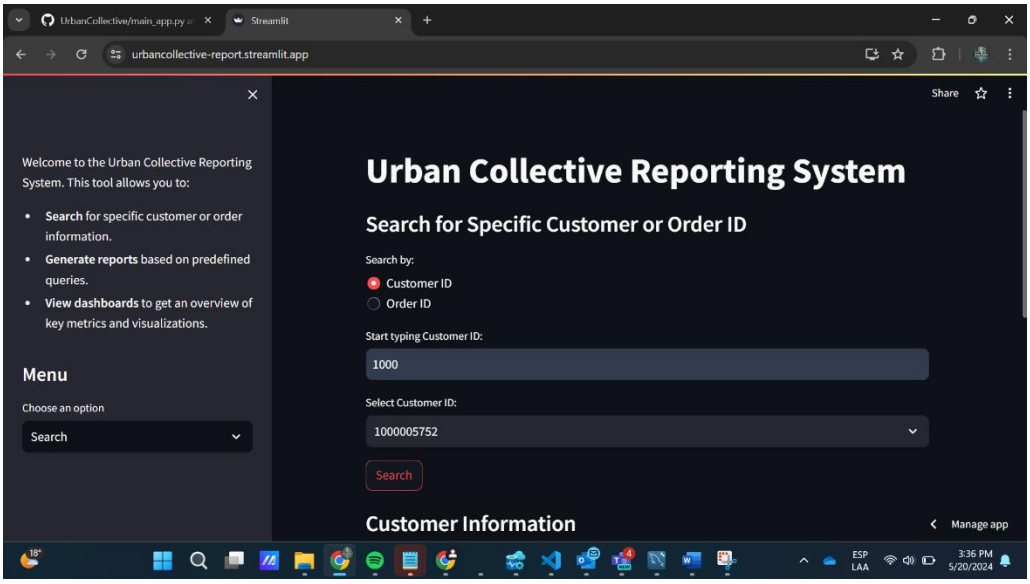
To create an interactive solution that interfaces with the Urbancollective Analytical Database, we leverage Python and Streamlit, a powerful framework for building data applications. The connection to our RDS database is established using the mysql.connector library, enabling secure and efficient data retrieval. Streamlit facilitates the creation of a dynamic web interface that allows users to execute predefined queries, visualize results, and interact with the data. The functionality is built around key modules such as pandas for data manipulation, seaborn and matplotlib for data visualization, and Streamlit's own components for user input and output.

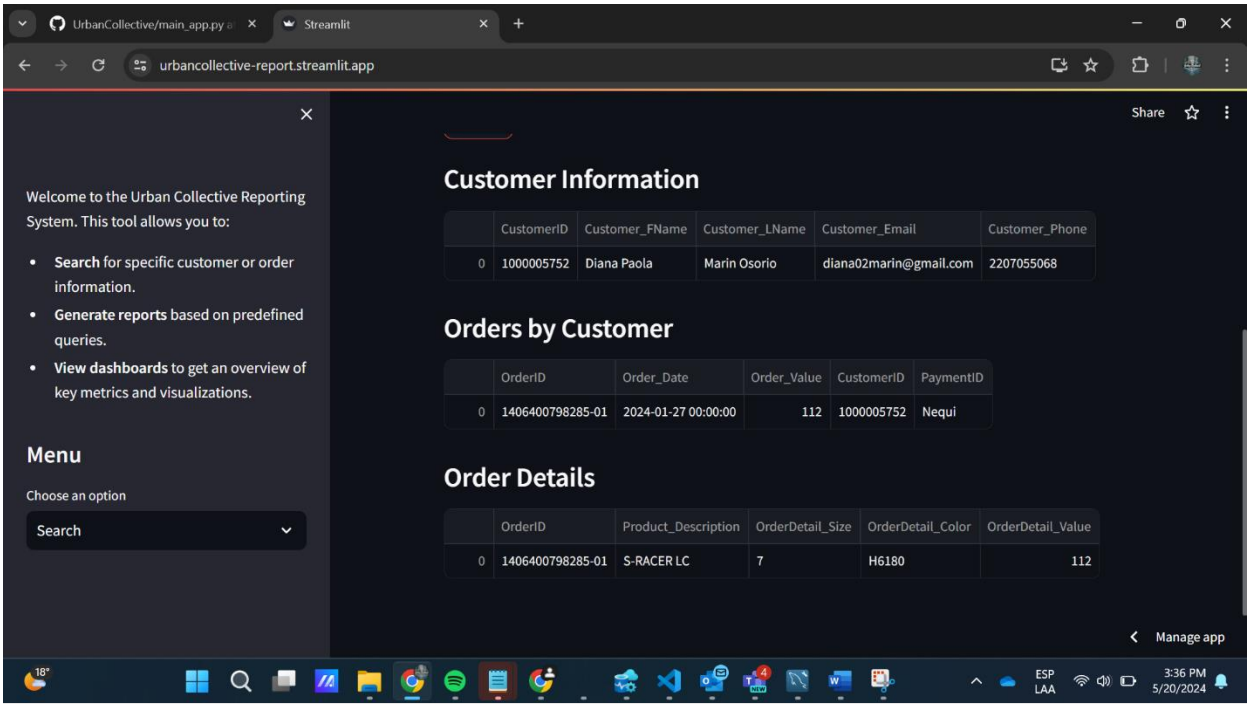
The provided Python script illustrates how these components work together to create a comprehensive reporting system. The script establishes a connection to the Urbancollective Analytical Database, defines SQL queries for specific reports, and executes these queries based on user input. Users can interact with the application through a Streamlit-based interface to search for specific customer or order information, run predefined queries, and view various dashboards. For instance, the dashboard provides an overview of key metrics like total customers, total orders, and revenue, while visualizations such as monthly revenue trends and top products by revenue offer deeper insights. This interactive solution demonstrates our ability to connect to and interact with the Urbancollective database using Python, providing a robust

platform for data analysis and reporting. Our reporting system is designed to handle dynamic queries that take user-defined parameters to generate customized results. For instance, we have a query that retrieves products with above-average ratings, filtered by a minimum number of reviews specified by the user. Another query calculates the Customer Lifetime Value (CLV) for the top 10 highest-spending customers, filtered by a minimum number of orders set by the user. These queries are executed by accepting parameters through the Streamlit interface, which are then passed to the SQL queries. This flexibility allows users to tailor their reports and analyses according to specific criteria, ensuring that the results are both relevant and insightful. By utilizing parameters in our queries, we enhance the interactivity and usability of our data application, making it a powerful tool for data-driven decision-making.

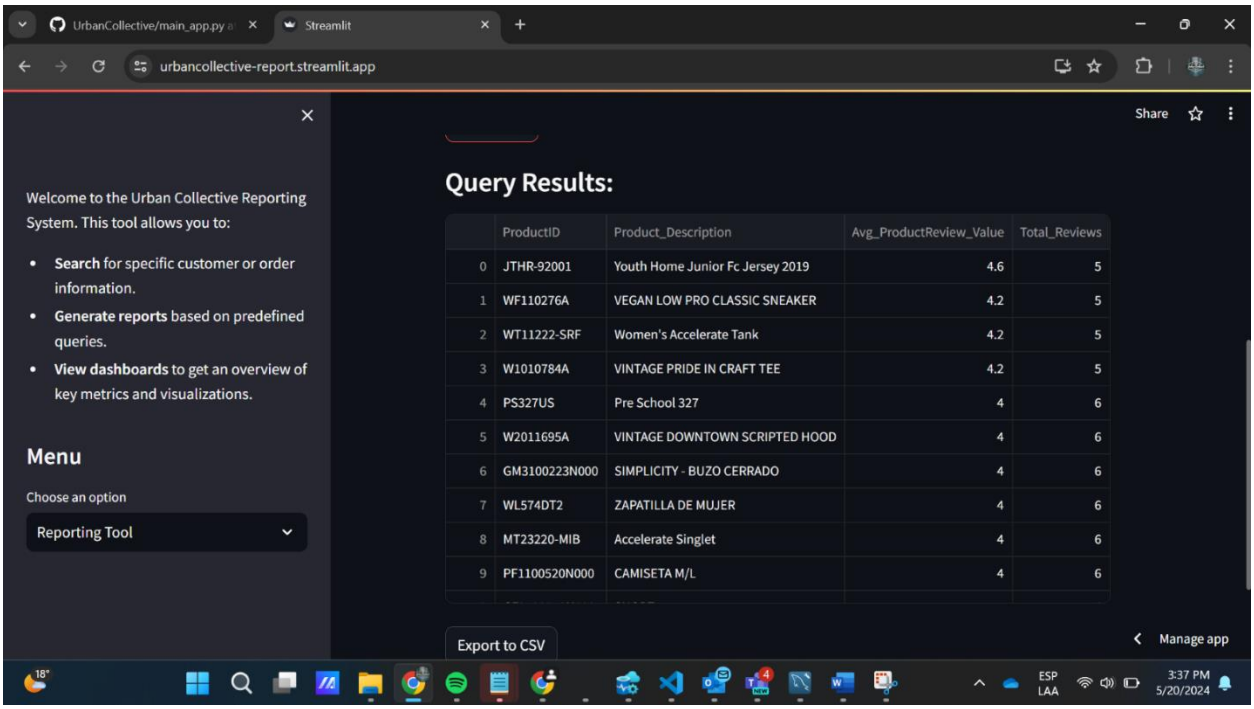
We will be working further on this Python interaction by making better visualizations in dashboard in future.

You can access this app using this link: https://urbancollective-report.streamlit.app/

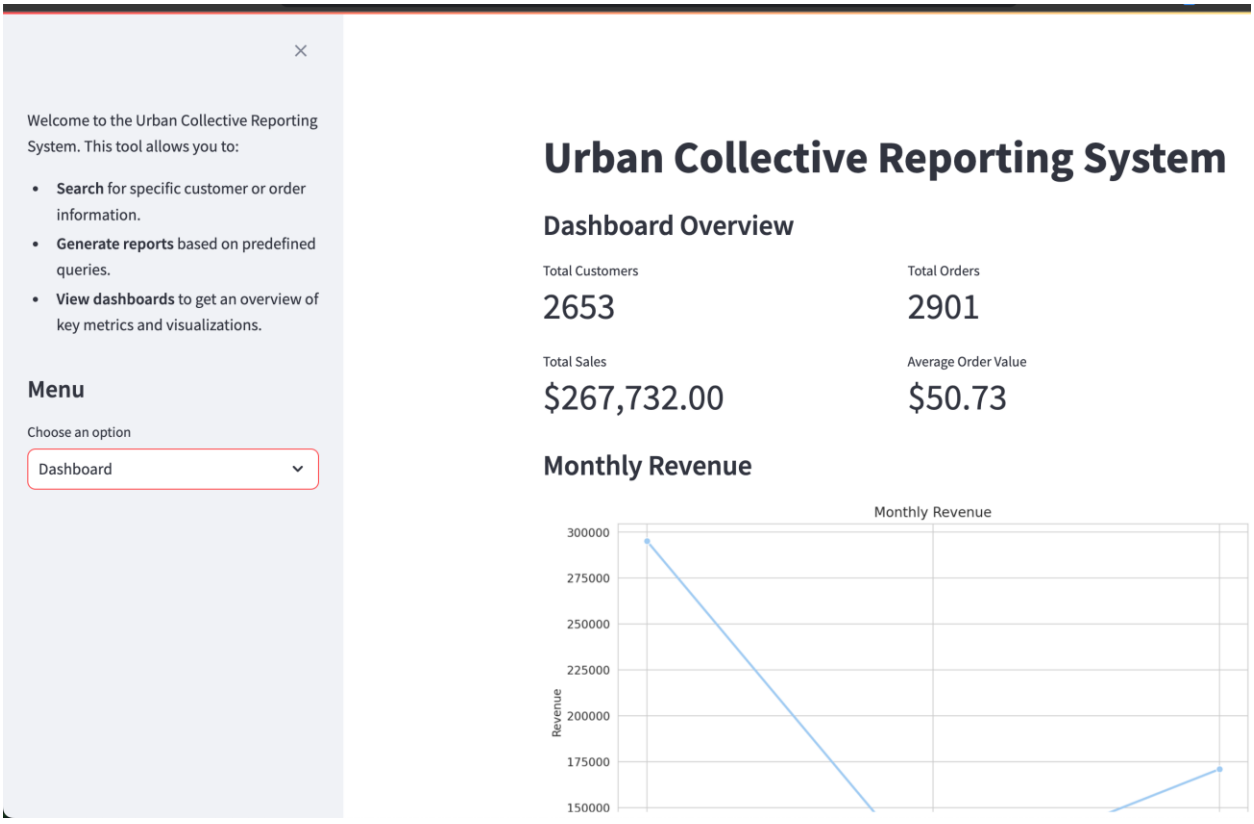



UrbanCollective/main_app.py
Streamlit
urbancollective-report.streamlit.app
Share
Welcome to the Urban Collective Reporting System. This tool allows you to:
Search for specific customer or order information.
Generate reports based on predefined queries.
View dashboards to get an overview of key metrics and visualizations.
Menu
Choose an option
Search
Customer Information
CustomerID Customer_FName Customer_LName Customer_Email Customer_Phone
0 1000005752 Diana Paola Marin Osorio diana02marin@gmail.com 2207055068
Orders by Customer
OrderID Order_Date Order_Value CustomerID PaymentID
0 1406400798285-01 2024-01-27 00:00:00 112 1000005752 Nequi
Order Details
OrderID Product_Description OrderDetail_Size OrderDetail_Color OrderDetail_Value
0 1406400798285-01 S-RACER LC 7 H6180 112
Manage app
ESP LAA
3:36 PM 5/20/2024


UrbanCollective/main_app.py
Streamlit
urbancollective-report.streamlit.app
Share
Welcome to the Urban Collective Reporting System. This tool allows you to:
Search for specific customer or order information.
Generate reports based on predefined queries.
View dashboards to get an overview of key metrics and visualizations.
Menu
Choose an option
Reporting Tool
Urban Collective Reporting System
Reporting Tool
1: Products with above-average ratings and a minimum number of reviews defined by the user
2: Customer Lifetime Value (CLV) for top 10 highest-spending customers with at least a number of orders defined by the user
Enter the query number:
1
Name: 1
Description: Products with above-average ratings and a minimum number of reviews defined by the user
Enter value for Minimum Reviews:
4
Manage app
ESP LAA
3:37 PM 5/20/2024



	ProductID	Product_Description	Avg_ProductReview_Value	Total_Reviews
0	JTHR-92001	Youth Home Junior Fc Jersey 2019	4.6	5
1	WF110276A	VEGAN LOW PRO CLASSIC SNEAKER	4.2	5
2	WT11222-SRF	Women's Accelerate Tank	4.2	5
3	W1010784A	VINTAGE PRIDE IN CRAFT TEE	4.2	5
4	PS327US	Pre School 327	4	6
5	W2011695A	VINTAGE DOWNTOWN SCRIPTED HOOD	4	6
6	GM3100223N000	SIMPLICITY - BUZO CERRADO	4	6
7	WL574DT2	ZAPATILLA DE MUJER	4	6
8	MT23220-MIB	Accelerate Singlet	4	6
9	PF1100520N000	CAMISETA M/L	4	6



Urban Collective Tableau Dashboard

Data visualization is done with the help of Tableau connected with the AWS Cloud database of urbancollective. This will help anyone to gather insights directly by pulling real-time data and analyzing the orders, products, customer details, product review, customer feedback and order value.

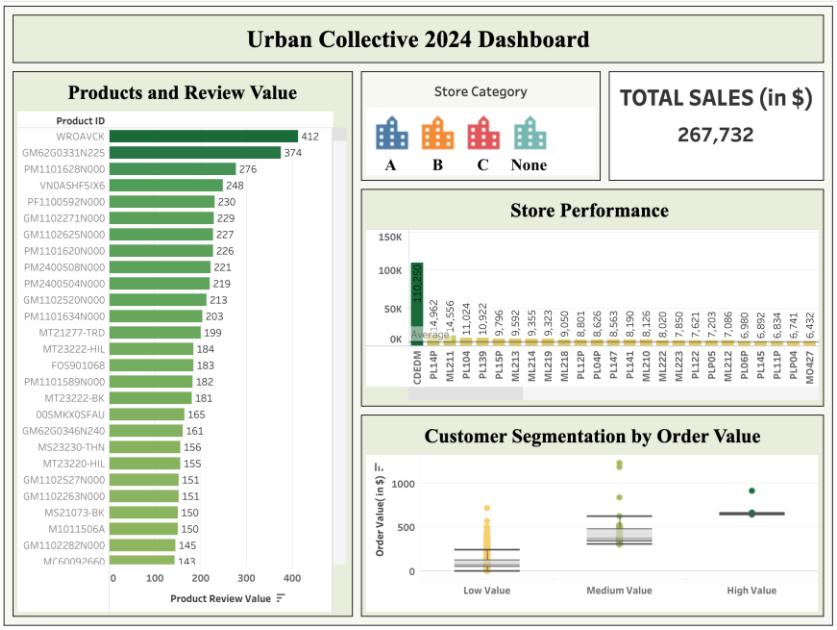
Note: The dashboard also consists of Store category ‘None’ which consists of orders which are not yet categorized.

Tableau Public Dashboard Link:

https://public.tableau.com/app/profile/amrapali.samanta4121/viz/UrbanCollective2024Dashboard_17168737322560/Dashboard1?publish=yes

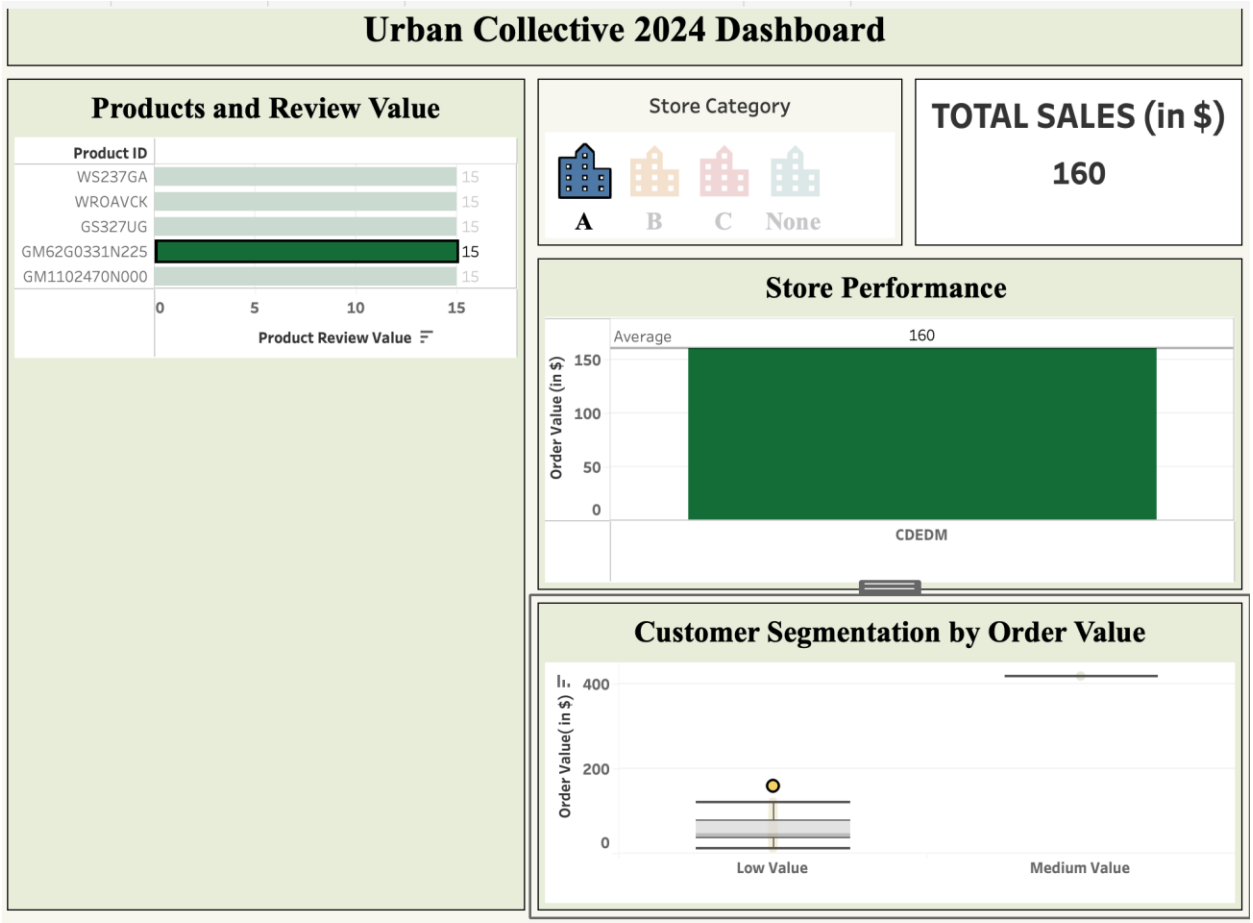
User Guide:

1. Select an option from Store A, B, or C to view updates on Total Sales.
2. Navigate to the Product Review section, select the Product ID bar, and observe changes across the entire dashboard data.
3. Navigate to Store performance, select a StoreID, and observe all changes. Also observe Average for any type of selection.
4. Navigate to the Customer segment section, select any order value from low, mid and high section and observe the product details and click product id to see store details related.
5. Refresh the page to quickly reset multiple filters



Example:

For below selections we select Store Category A, then Product ID, then a customer with low value order of 160 and the Store performance and Total Sales is updated.


Urban Collective 2024 Dashboard
Products and Review Value
Product ID
WS237GA
WROAVCK
GS327UG
GM62G0331N225
GM1102470N000
15
15
15
15
15
0
5
10
15
Product Review Value
Store Category
A
B
C
None
TOTAL SALES (in $)
160
Store Performance
Average
160
Order Value (in $)
150
100
50
0
CDEDM
Customer Segmentation by Order Value
Order Value(in $)
400
200
0
Low Value
Medium Value

Appendix:

Data Dictionary:

- **Customer**

Field	Type	Description
CustomerID	VARCHAR(45)	Unique identifier for each customer.
Customer_FName	VARCHAR(45)	First name of the customer.
Customer_LName	VARCHAR(45)	Last name of the customer.
Customer_Email	VARCHAR(45)	Email address of the customer.
Customer_Phone	VARCHAR(45)	Phone number of the customer.

- **Order**

Field	Type	Description
OrderID	VARCHAR(50)	Unique identifier for each order.
Order_Date	DATETIME	Date and time when the order was placed.
Order_Value	INT	Total value of the order.
CustomerID	VARCHAR(45)	Identifier linking the order to the customer who placed it.
PaymentID	VARCHAR(50)	Identifier for the payment associated with the order.

- **OrderDetail**

Field	Type	Description
OrderDetail_Quantity	INT	Quantity of a specific product in an order.
OrderDetail_Value	INT	Total value of a specific product in an order.
OrderDetail_Size	VARCHAR(45)	Size of the product in the order.
OrderDetail_Color	VARCHAR(45)	Color of the product in the order.
OrderID	VARCHAR(50)	Identifier linking the order detail to the order.
ProductID	VARCHAR(45)	Identifier linking the order detail to the product.

StoreID	VARCHAR(6)	Identifier linking the order detail to the store.

- **Payment**

Field	Type	Description
PaymentID	VARCHAR(50)	Unique identifier for each payment.
Payment_Description	VARCHAR(45)	Description of the payment.

- **Product**

Field	Type	Description
ProductID	VARCHAR(45)	Unique identifier for each product.
Product_Description	VARCHAR(100)	Description of the product.
Product_Brand	VARCHAR(45)	Brand name of the product.
Product_Gender	VARCHAR(45)	Gender for which the product is intended.
ProductTypeID	INT	Identifier linking the product to its product type.

- **ProductReview**

Field	Type	Description
ProductReview_Date	DATETIME	Date and time when the product review was submitted.
ProductReview_Value	INT	Rating or value assigned to the product review.
OrderID	VARCHAR(50)	Identifier linking the product review to the order.

ProductID	VARCHAR(45)	Identifier linking the product review to the product.

- **ProductType**

Field	Type	Description
ProductTypeID	INT	Unique identifier for each product type.
ProductType_Description	VARCHAR(45)	Description of the product type.

- **Store**

Field	Type	Description
StoreID	VARCHAR(6)	Unique identifier for each store.
Store_Description	VARCHAR(45)	Description of the store.
Store_Category	CHAR(1)	Category of the store.