



Data Science Immersive

Etinosa Ekomwenrenren

SQL Final Project

October 2023

Abstract

A total of 5 csv data files were provided which contain raw data extracted from an e-commerce analytics tool. The essence of the project was to load that raw data into a database, understand the context for the attributes in the data, clean the data using SQL concepts and queries studied in the course, and then finally glean insights into the operations of the business from the clean and processed data. Also important is to understand and utilize QA processes throughout the aforementioned stages.

Contents

1	Introduction	1
1.1	Overview	1
1.2	Introducing the dataset	2
1.2.1	The all sessions data file	2
1.2.2	The products data file	2
1.2.3	The sales report data file	3
1.2.4	The sales by sku data file	4
1.2.5	The analytics data file	4
2	Data Cleaning	6
2.1	Overview	6
2.2	Cleaning the data	6
2.2.1	The all sessions dataset	7
2.2.2	The products, sales reports, and sales by sku datasets	15
2.2.3	The analytics dataset	16
3	Starting with questions	19
3.1	Question 1: Which cities and countries have the highest level of transaction revenues on the site?	19
3.2	Question 2: What is the average number of products ordered from visitors in each city and country?	21
3.3	Question 3: Is there any pattern in the types (product categories) of products ordered from visitors in each city and country?	24

3.4	Question 4: What is the top-selling product from each city/country? Can we find any pattern worthy of noting in the products sold?	27
4	ERD for database	31
5	Starting with data	32
5.1	Question 1: What products are among the top 10 sold but whose stock level are currently below the average stock level for all products?	32

List of Figures

1.1	Subsection of rows and columns in the all sessions data file.	2
1.2	Subsection of rows and columns in the products data file.	3
1.3	Subsection of rows and columns in the sales report data file.	3
1.4	Subsection of rows and columns in the sales by sku data file.	4
1.5	Subsection of rows and columns in the analytics data file.	5
3.1	Countries with the highest level of transaction revenues on site.	20
3.2	Cities with the highest level of transaction revenues on site.	21
3.3	Countries with the highest level of average number of products ordered on site. .	23
3.4	Cities with the highest level of average number of products ordered on site. . . .	24
3.5	Countries together with the most popular product category on site.	26
3.6	Cities together with the most popular product category on site.	27
3.7	Countries together with the most popular product sold on site.	29
3.8	Cities together with the most popular product sold on site.	30
4.1	ERD for database	31
5.1	Countries together with the most popular product sold on site.	33
5.2	Cities together with the most popular product sold on site.	34

List of Tables

Listings

2.1	Sample SQL query for the products table creation	6
2.2	Sample SQL query for verifying rows after data importation	7
2.3	Sample SQL query for the all sessions table cleaning	7
2.4	Sample SQL query for verifying rows after data transformation	9
2.5	Sample SQL query for verifying dependency of attributes	10
2.6	Sample UDF for verifying dependency of attributes	10
2.7	Sample UDF for verifying eligibility of a column as a PK	11
2.8	Sample SQL query for verifying replication of content between tables	12
2.9	Sample SQL query for human validation of one table being a subset of another .	13
2.10	Sample SQL query for checking if ‘transactionrevenue’ is a subset of ‘totaltrans- actionrevenue’	13
2.11	Sample UDF for verifying if a column is a subset of another within the same table	14
2.12	Sample SQL query for inspecting duplicate attributes across tables	15
2.13	Sample SQL query for creating the product info and sales info tables	16
2.14	Sample SQL query for cleaning the analytics table	17
2.15	Sample SQL query for inspecting duplicate attributes across tables	18
3.1	Sample SQL query for ranking countries based on transaction revenue	19
3.2	Sample SQL query for inspecting duplicate attributes across tables	20
3.3	Sample SQL query for ranking countries based on transaction revenue	21
3.4	Sample SQL query for inspecting duplicate attributes across tables	22
3.5	Sample SQL query for ranking countries based on transaction revenue	24
3.6	Sample SQL query for inspecting duplicate attributes across tables	25
3.7	Sample SQL query for ranking countries based on transaction revenue	27
3.8	Sample SQL query for inspecting duplicate attributes across tables	28

5.1	Sample SQL query for ranking countries based on transaction revenue	32
-----	---	----

Chapter 1

Introduction

The primary objective of this project is to solidify the SQL concepts introduced during the first two weeks of this course by engaging with raw data sourced from five data files.

1.1 Overview

The dataset offers insights into each visitor's interaction with products on the ecommerce platform. It details information such as the date and time of the visit, visitor's country and city, page views, duration of the visit, and any transactions that took place. Furthermore, the dataset provides sales figures, revenue data, and product specifics related to the ecommerce site.

The initial step involves importing the five CSV files into distinct tables within a PostgreSQL database named *ecommerce*. Once uploaded, it's crucial to grasp the data's context and significance. This phase also includes identifying potential challenges like data type inconsistencies, missing or inconsistent records, scale discrepancies, and unit misrepresentations. Subsequently, these problems are addressed using SQL queries to clean or preprocess the data.

Upon refining the dataset, preliminary insights can be derived by considering questions such as:

1. Which cities and countries generate the most transaction revenues for the site?
2. What's the average number of products purchased by visitors in each city and country?
3. Are there discernible patterns in the product categories selected by visitors from various cities and countries?

4. Which product is the best-seller in each city or country? Are there any notable trends in the products sold?
5. Can the revenue contributions from each city or country be summarized?

While addressing these questions sheds light on the dataset, deeper insights can still be unearthed with more extensive analysis.

Throughout each step mentioned above, it's vital to verify hypotheses and ensure the accuracy of findings. Therefore, a comprehensive set of queries, underpinned by a robust QA process, should be executed concurrently with these tasks.

1.2 Introducing the dataset

1.2.1 The all sessions data file

A sample subsection of the columns and rows of the CSV file is shown in the image below

	A	B	C	D	E	F	G	H	I	J	K	
1	fullVisitorId	channelGrouping	time	country	city	totalTransactionRevenue	transactions	timeOnSite	pageviews	sessionQualityDim	date	visitId
2	2817722496551184	Direct	122213	Taiwan	(not set)			142	7		20160913	f
3	6938960486452471	Referral	268947	United States	not available in demo dataset			269	6		20170421	f
4	5307554331754279	Organic Search	61402	United States	not available in demo dataset			61	6		20170312	f
5	5256949873742090	Display	8655	United States	not available in demo dataset			96	5		20170215	f
6	7549308697995967	Referral	0	United States	London			1238	3		20161218	f
7	5444913961890272	Organic Search	25667	(not set)	(not set)			33	4		20160911	f
8	9147600624562955	Organic Search	14737	El Salvador	not available in demo dataset			116	6		20170413	f
9	4476811245637991	Organic Search	123686	United States	not available in demo dataset			124	8		20160902	f
10	6153038635419747	Organic Search	8860	United Kingdom	not available in demo dataset			9	2		20170628	f
11	0719635766852234	Organic Search	39393	Australia	Sydney			39	2		20170613	f
12	3311454414003524	Organic Search	0	United States	Philadelphia				1	1	20170718	f
13	9242525754496359	Organic Search	0	Serbia	not available in demo dataset				1		20160827	f
14	9063370709205682	Organic Search	0	Canada	not available in demo dataset				1	1	20170704	f
15	0271596310534061	Organic Search	64390	Germany	not available in demo dataset			64	3		20161121	f
16	0209475620744522	Organic Search	0	United Kingdom	not available in demo dataset				1		20170301	f
17	7686785537772322	Paid Search	26488	United States	not available in demo dataset			59	5		20170424	f
18	0792073733957836	Organic Search	0	Germany	not available in demo dataset				1		20170123	f
19	7894637781859655	Direct	25112	United States	Sunnyvale			144	4		20170209	f
20	3921213616472444	Referral	4046	United States	Mountain View			4	2		20160902	f

Figure 1.1: Subsection of rows and columns in the all sessions data file.

1.2.2 The products data file

A sample subsection of the columns and rows of the CSV file is shown in the image below

	A	B	C	D	E	F	G
1	SKU	name	orderedQuantity	stockLevel	restockingLeadTime	sentimentScore	sentimentMagnitude
2	GGADFBBSBKS4234	PC gaming speakers	0	100	1		
3	GGOEGAAX0581	Women's Colorblock	0	0	8	0.8	2
4	GGOEGAAX0596	Men's Quilted Insula	26	32	8	0.8	2
5	GGOEGAAX0365	Women's Scoop Nec	65	73	6	0	0.5
6	GGOEGEV070899	Bongo Cupholder Bl	85	115	8	0	0.5
7	GGOEGESC014699	Aluminum Handy En	66	90	9	0	0.5
8	GGOEGAAX0325	Men's Short Sleeve F	53	104	12	0	0.5
9	GGOEGAAX0296	Women's Short Sleev	19	33	15	0	0.5
10	GGOEGHGH019699	Sunglasses	1573	2086	6	0.5	0.5
11	GGOEGDWR015799	Red Shine 15 oz Mu	1058	1417	20	0.3	0.5
12	GGOEGAAX0081	Recycled Paper Jou	545	740	5	0.3	0.5
13	GGOEGALB036514	Women's Scoop Nec	116	154	5	0.3	0.5
14	GGOEGAWH061350	Onesie Green	7	9	5	0.3	0.5
15	GGOENEBQ084699	Leaming Thermostat	849	1142	5	0.3	0.5
16	GGOEAHPB076114	Android Stretch Fit F	6	8	5	0.3	0.5
17	GGOEGADJ059715	Men's Quilted Insula	30	39	5	0.3	0.5
18	GGOEWAEA083899	Dress Socks	15	20	6	0.3	0.5
19	9180850	Ballpoint Stylus Pen	0	0	6	0.3	0.5
20	9183118	Women's Fleece Hoc	0	0	6	0.3	0.5
21	GGOEYDHJ056099	22 oz Bottle Infuser	1465	1944	7	0.3	0.5

Figure 1.2: Subsection of rows and columns in the products data file.

1.2.3 The sales report data file

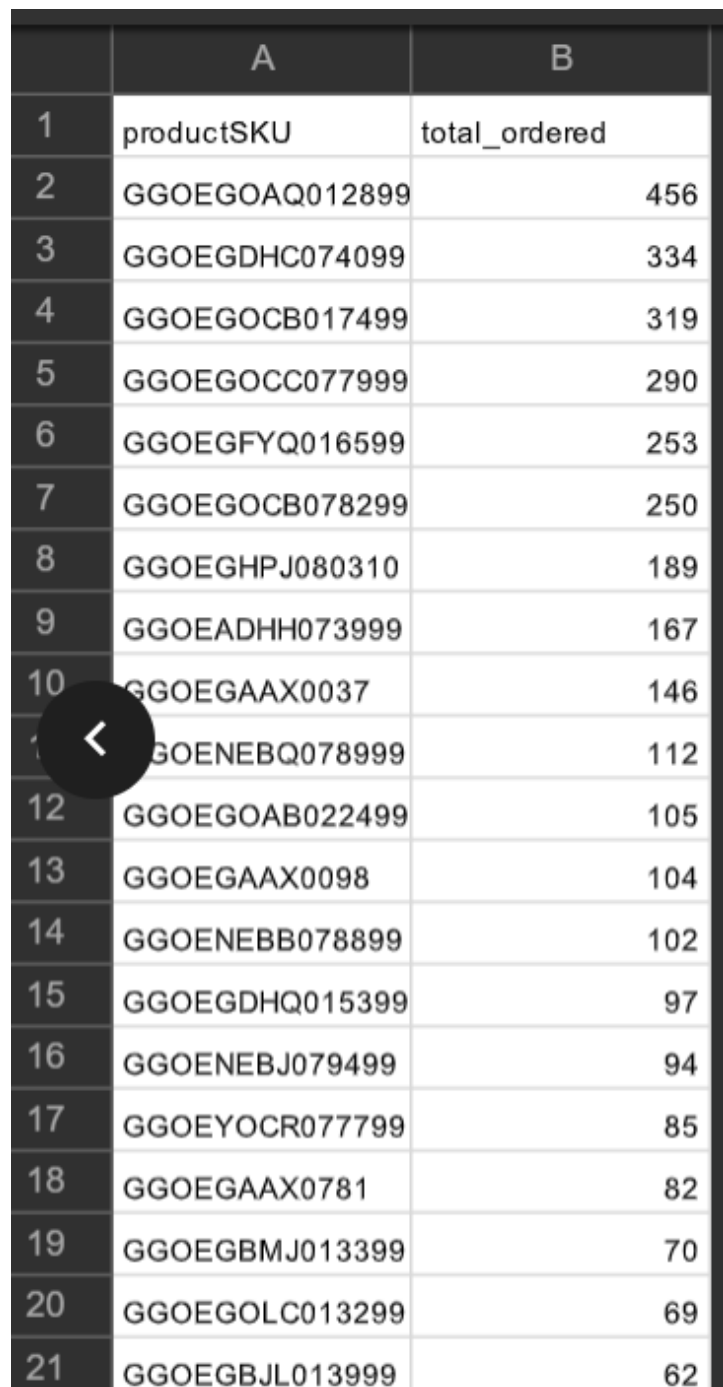
A sample subsection of the columns and rows of the CSV file is shown in the image below

	A	B	C	D	E	F	G	H
1	productSKU	total_ordered	name	stockLevel	restockingLeadTime	sentimentScore	sentimentMagnitude	ratio
2	GGOEGAAX0081	42	Recycled Paper Jou	740	5	0.3	0.5	0.05675675676
3	GGOENEBQ084699	7	Leaming Thermostat	1142	5	0.3	0.5	0.006129597198
4	GGOEGAAX0325	6	Men's Short Sleeve F	104	12	0	0.5	0.05769230769
5	GGOEYDHJ056099	50	22 oz Bottle Infuser	1944	7	0.3	0.5	0.02572016461
6	GGOEADHH073999	167	Android 17oz Stainle	283	8	0.3	0.5	0.5901060071
7	GGOEWAEA083899	1	Dress Socks	20	6	0.3	0.5	0.05
8	GGOEGAAJ032313	6	Men's Short Sleeve F	61	8	0.3	0.5	0.09836065574
9	GGOEGAA033815	9	Men's Vintage Badge	578	13	0.3	0.5	0.01557093426
10	GGOEGPJC203399	14	Crunch Noise Dog Tc	352	9	0.7	0.5	0.03977272727
11	GGOEGALB034113	16	Women's Vintage He	153	14	0.3	0.5	0.1045751634
12	GGOEGHGH019699	14	Sunglasses	2086	6	0.5	0.5	0.006711409396
13	GGOENEBJ079499	94	Leaming Thermostat	2525	7	0.3	0.5	0.03722772277
14	GGOEGOBG023599	3	Colored Pencil Set	367	13	0.3	0.5	0.008174386921
15	GGOEGAAX0620	20	Infant Short Sleeve T	52	7	0.3	0.5	0.3846153846
16	GGOEGDWR015799	5	Red Shine 15 oz Mu	1417	20	0.3	0.5	0.00352858151
17	GGOEGODR017799	39	Recycled Mouse Pac	1756	39	0.3	1	0.0222095672
18	GGOEGBJC014399	3	Tote Bag	155	6	0.6	1	0.01935483871
19	GGOENEBQ078999	112	Cam Outdoor Securi	4683	10	0.6	1	0.02391629297
20	GGOEGAEQ027913	2	Women's Short Sleev	65	7	0.6	1	0.03076923077
21	GGOEGAAX0795	7	25L Classic Rucksac	114	6	0.7	1	0.06140350877
22	GGOEGGCX056199	1	Gift Card- \$100.00	77	10	0.6	1	0.01298701299

Figure 1.3: Subsection of rows and columns in the sales report data file.

1.2.4 The sales by sku data file

A sample subsection of the columns and rows of the CSV file is shown in the image below



	A	B
1	productSKU	total_ordered
2	GGOEGOAQ012899	456
3	GGOEGDHC074099	334
4	GGOEGOCB017499	319
5	GGOEGOCC077999	290
6	GGOEGFYQ016599	253
7	GGOEGOCB078299	250
8	GGOEGHPJ080310	189
9	GGOEADHH073999	167
10	GGOEGAAX0037	146
11	GGOENEBQ078999	112
12	GGOEGOAB022499	105
13	GGOEGAAX0098	104
14	GGOENEBB078899	102
15	GGOEGDHC015399	97
16	GGOENEBJ079499	94
17	GGOEYOCR077799	85
18	GGOEGAAX0781	82
19	GGOEGBMJ013399	70
20	GGOEGOLC013299	69
21	GGOEGBJL013999	62

Figure 1.4: Subsection of rows and columns in the sales by sku data file.

1.2.5 The analytics data file

A sample subsection of the columns and rows of the CSV file is shown in the image below

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	
1	visitNumb	visitId	visitStartT	date	fullvisitorId	userId	channelGr	socialEnga	units_sold	pageviews	timeonsite	bounces	revenue	unit_price	
2	7	1.5E+09	1.5E+09	20170625	9.44E+18		Display	Not Socially Engaged		1		1		8990000	
3	7	1.5E+09	1.5E+09	20170625	9.44E+18		Display	Not Socially Engaged		1		1		10990000	
4	7	1.5E+09	1.5E+09	20170625	9.44E+18		Display	Not Socially Engaged		1		1		10990000	
5	7	1.5E+09	1.5E+09	20170625	9.44E+18		Display	Not Socially Engaged		1		1		10990000	
6	7	1.5E+09	1.5E+09	20170625	9.44E+18		Display	Not Socially Engaged		1		1		11990000	
7	7	1.5E+09	1.5E+09	20170625	9.44E+18		Display	Not Socially Engaged		1		1		11990000	
8	7	1.5E+09	1.5E+09	20170625	9.44E+18		Display	Not Socially Engaged		1		1		16990000	
9	7	1.5E+09	1.5E+09	20170625	9.44E+18		Display	Not Socially Engaged		1		1		18990000	
10	7	1.5E+09	1.5E+09	20170625	9.44E+18		Display	Not Socially Engaged		1		1		18990000	
11	7	1.5E+09	1.5E+09	20170625	9.44E+18		Display	Not Socially Engaged		1		1		18990000	
12	7	1.5E+09	1.5E+09	20170625	9.44E+18		Display	Not Socially Engaged		1		1		22990000	
13	7	1.5E+09	1.5E+09	20170625	9.44E+18		Display	Not Socially Engaged		1		1		24990000	
14	1	1.5E+09	1.5E+09	20170704	9.25E+17		Direct	Not Socially Engaged		1		1		1.1E+08	
15	1	1.5E+09	1.5E+09	20170704	9.25E+17		Direct	Not Socially Engaged		1		1		1.05E+08	
16	1	1.5E+09	1.5E+09	20170704	9.25E+17		Direct	Not Socially Engaged		1		1		1.1E+08	
17	1	1.5E+09	1.5E+09	20170704	9.25E+17		Direct	Not Socially Engaged		1		1		1.1E+08	
18	1	1.5E+09	1.5E+09	20170704	9.25E+17		Direct	Not Socially Engaged		1		1		1.2E+08	
19	1	1.5E+09	1.5E+09	20170704	9.25E+17		Direct	Not Socially Engaged		1		1		1.2E+08	
20	34	1.5E+09	1.5E+09	20170625	5.02E+18		Organic Se	Not Socially Engaged		1		1		12990000	
21	34	1.5E+09	1.5E+09	20170625	5.02E+18		Organic Se	Not Socially Engaged		1		1		17990000	
22	34	1.5E+09	1.5E+09	20170625	5.02E+18		Organic Se	Not Socially Engaged		1		1		19990000	
23	34	1.5E+09	1.5E+09	20170625	5.02E+18		Organic Se	Not Socially Engaged		1		1		19990000	
24	34	1.5E+09	1.5E+09	20170625	5.02E+18		Organic Se	Not Socially Engaged		1		1		19990000	
25	34	1.5E+09	1.5E+09	20170625	5.02E+18		Organic Se	Not Socially Engaged		1		1		19990000	

Figure 1.5: Subsection of rows and columns in the analytics data file.

Chapter 2

Data Cleaning

2.1 Overview

In this chapter, we provide details on the procedures undergone when loading and cleaning the data, as well as the associated QA steps.

2.2 Cleaning the data

All CSV files are loaded into their respective tables in the *ecommerce* database. The Listing below provides the sample code for the creation of the respective tables in the *ecommerce* database. All columns are given the ‘text’ datatype for generalization, and all raw data is then imported into the created tables.

```
1 CREATE TABLE products
2 ( sku text,
3   name text,
4   orderedquantity text,
5   stocklevel text,
6   restockingleadtime text,
7   sentiment score text,
8   sentimentmagnitude text
9 );
```

Listing 2.1: Sample SQL query for the products table creation

QuestionIOS environment:

QA: Loading data into tables

It is important to verify that the total rows in the table matches with the rows in the original CSV files.

```
1 SELECT COUNT(*)
2 FROM all_sessions;
```

Listing 2.2: Sample SQL query for verifying rows after data importation

2.2.1 The all sessions dataset

Cleaning procedure

The columns in the all sessions dataset are preprocessed in one go to correct issues such as choosing the correct datatype. For columns such as ‘country’, ‘city’, and ‘productvariant’, rows containing missing data indicated by *(not set)* or *not available in demo dataset* are converted to NULL. This choice is made because the NULL datatype more accurately reflects the states of the rows and the treatment of NULLs in SQL is standardized and feature-rich. Furthermore, columns containing price information such as ‘totaltransactionrevenue’, ‘producprice’, ‘transactionrevenue’, and ‘productrevenue’ had their scales corrected by dividing by 1×10^6 . A new table is created with the preprocessed/cleaned data. The SQL query for this cleaning is provided below.

```
1 CREATE TABLE temp_session_info as (
2 SELECT
3     fullvisitorid,
4     channelgrouping,
5     time::int,
6     (CASE
7     WHEN country LIKE '(%)' OR country LIKE 'not%' THEN NULL
8     ELSE country
9     END
10 ) as country,
11     (CASE
12     WHEN city LIKE '(%)' OR city LIKE 'not%' THEN NULL
13     ELSE city
```

```
14  END
15  ) as city,
16  (totaltransactionrevenue::real)/1000000 as totaltransactionrevenue,
17  transactions::real,
18  timeonsite::int,
19  pageviews::int,
20  sessionqualitydim::int,
21  date::date,
22  visitid,
23  type,
24  productrefundamount,
25  productquantity::int,
26  (productprice::real)/1000000 as productprice,
27  (productrevenue::real)/1000000 as productrevenue,
28  productsku,
29  v2productname,
30  v2productcategory,
31  (CASE
32  WHEN productvariant LIKE '(' OR productvariant LIKE 'not%' THEN NULL
33  ELSE productvariant
34  END
35  ) as productvariant,
36  currencycode,
37  itemquantity::int,
38  (itemrevenue::real)/1000000 as itemerevenue,
39  (transactionrevenue::real)/1000000 as transactionrevenue,
40  transactionid,
41  pagetitle,
42  searchkeyword,
43  pagepathlevel1,
44  ecommerceactiontype::int,
45  ecommerceactionstep::int,
46  ecommerceactionoption
47  FROM all_sessions);
```

Listing 2.3: Sample SQL query for the all sessions table cleaning

QA: Cleaning the all sessions dataset

It is important to verify that the total rows in the new table match the rows in the original table after the transformations. Additionally, it is good practice to confirm that the numerical values after the scale correction remain consistent. A sample SQL query for the ‘totaltransactionrevenue’ column is provided below.

```
1 SELECT
2 SUM(totaltransactionrevenue::numeric) AS total_revenue,
3 'all_sessions table' AS type
4 FROM all_sessions
5
6 UNION ALL
7
8 SELECT
9 SUM(totaltransactionrevenue*1000000) AS total_revenue,
10 'new table' AS type
11 FROM temp_session_info;
```

Listing 2.4: Sample SQL query for verifying rows after data transformation

Understanding the dataset

- First, we try to determine the granularity of the table. What does an individual row from the table mean? In exploring this question, it is observed that the table does not seem normalized: groups of non-key attributes seem more dependent on other ‘major’ attributes. Examples of these ‘major’ attributes are visitor information and product information
- However, it is important to validate any such hypothesis or observation before acting. For example, one hypothesis is that attributes like ‘country’, ‘city’ and ‘currencycode’ depend more directly on the ‘fullvisitorid’ than on other attributes like the particular session represented as ‘vistid’ in the table. Attributes like ‘country’ and ‘city’ seem self-evident. But for an attribute like ‘currencycode’, we might need to be more careful.

QA: Understanding the all sessions dataset

The following SQL query helps us QA the process of confirming that ‘currencycode’ indeed is unique to the ‘fullvisitorid’. An empty row from the below query shows that no one visitor has multiple currencies associated with it from the entire all sessions table.

```

1 select fullvisitorid, count(distinct currencycode)
   as freq
2 from all_sessions
3 group by fullvisitorid
4 having count(distinct currencycode) > 1
5 order by freq desc;

```

Listing 2.5: Sample SQL query for verifying dependency of attributes

- The code confirms that ‘currencycode’ is indeed dependent on the ‘fullvisitorid’ column.

We can generalize this check using the user-defined function below.

```

1 CREATE OR REPLACE FUNCTION is_attribute_unique(
2     p_table_name TEXT,
3     p_main_attribute TEXT,
4     p_dependent_attribute TEXT
5 ) RETURNS TEXT AS $$
6 DECLARE
7     v_count INTEGER;
8     v_query TEXT;
9 BEGIN
10    v_query := format('
11        SELECT COUNT(*)
12        FROM (
13            SELECT %I, COUNT(DISTINCT %I) as freq
14            FROM %I
15            GROUP BY %I
16            HAVING COUNT(DISTINCT %I) > 1
17        ) sub_query
18    ', p_main_attribute, p_dependent_attribute, p_table_name,

```

```

    p_main_attribute, p_dependent_attribute);
19
20 EXECUTE v_query INTO v_count;
21
22 IF v_count = 0 THEN
23     RETURN 'unique';
24 ELSE
25     RETURN 'not unique';
26 END IF;
27 END;
28 $$ LANGUAGE plpgsql;

```

Listing 2.6: Sample UDF for verifying dependency of attributes

- Unexpectedly, using similar logic, ‘v2productname’ is not unique to ‘productsku’. Similarly, ‘productvariant’, ‘productprice’ and ‘productcategory’ are not unique to ‘productsku’. Additionally, it was discovered that a ‘visitid’ can have more than one ‘fullvisitorid’.
- With the foregoing, the conclusion is that more domain information is required to fully normalize the all sessions table. For example, what does it mean, precisely, for a single ‘visitid’ to have more than 1 distinct ‘fullvisitorid’?
- Next, we need to determine if a primary key already exists in the table. We do this by checking if any of the column entries contain only unique values. The following UDF provides a generalized way to check if all entries of a particular column are unique in the table.

```

1 CREATE OR REPLACE FUNCTION check_unique_values(table_name text, column_name
    text)
2 RETURNS BOOLEAN AS $$
3 DECLARE
4     total_count INTEGER;
5     unique_count INTEGER;
6     query_text TEXT;
7 BEGIN
8     -- Get total count
9     query_text := format('SELECT COUNT(%I) FROM %I', column_name,
    table_name);
10 EXECUTE query_text INTO total_count;
11

```

```

12  -- Get distinct count
13  query_text := format('SELECT COUNT(DISTINCT %I) FROM %I', column_name,
14                        table_name);
15
16  EXECUTE query_text INTO unique_count;
17
18  -- Return if both counts are the same (column has only unique values)
19  RETURN total_count = unique_count;
20
21 END;
22
23 $$ LANGUAGE plpgsql;

```

Listing 2.7: Sample UDF for verifying eligibility of a column as a PK

- The conclusion of the previous check is that no individual candidate column exists in the table. In fact, the combination of ‘visitid’, ‘fullvisitorid’, and ‘productsku’ do not qualify as a composite key. This observation is related to the earlier conclusion that more information is required to determine the granularity of the all sessions table. In that case, let’s create a new PK. Since the all sessions table does not contain duplicate rows, we can create a PK from the row numbers using the ROW_NUMBER() function, if desired.
- The all sessions table contains two columns containing revenue information: ‘totaltransactionrevenue’ and the ‘transactionrevenue’. To determine if the information is possibly replicated in both tables, the following query is utilized.

```

1  select fullvisitorid as id,
2         totaltransactionrevenue as tot_trans_rev,
3         transactions as trans,
4         transactionrevenue as trans_rev
5  from all_sessions
6  where totaltransactionrevenue is not null and
7         transactionrevenue is not null;

```

Listing 2.8: Sample SQL query for verifying replication of content between tables

- The above query showed that when values are present in both columns, they capture the same information. Now, we need to decide how to proceed. Should we drop one column? If yes, which of the columns? Does any column potentially contain information not present in another? My hypothesis is that the ‘transactionrevenue’ column is a subset of the ‘totaltransactionrevenue’ column. This can be validated by manual human inspection using the following query:

```
1  select fullvisitorid as id,
2      totaltransactionrevenue as tot_trans_rev,
3      transactions as trans,
4      transactionrevenue as trans_rev
5  from all_sessions
6  where totaltransactionrevenue is not null or
7      transactionrevenue is not null;
```

Listing 2.9: Sample SQL query for human validation of one table being a subset of another

- The result from the above query confirms my hypothesis. However, is there any automated way to confirm this? For example, a query where the end result shows yes/no or pass/fail answer?
- The following encapsulates my logic for an automated test.

Problem Statement: Given column A and column B in a table with a unique primary key, determine if column B is a subset of column A for non null values.

The idea is to check the rows using column A and the primary key and union with the rows of column B with the primary key. If the result of the union is equal to the number of rows using only column A and the primary key, then column B is a subset of column A. The following SQL query provides the check for 'totaltransactionrevenue' and 'transactionrevenue'.

```
1  select index_id as id,
2      totaltransactionrevenue as revenue
3  from session_info
4  where totaltransactionrevenue is not null
5
6  union
7
8  select index_id as id,
9      transactionrevenue as revenue
10 from session_info
11 where transactionrevenue is not null;
```

Listing 2.10: Sample SQL query for checking if 'transactionrevenue' is a subset of 'totaltransactionrevenue'

- The resulting check confirms the hypothesis. As a result, only the 'totaltransactionrev-

enue' column will be considered going forward. The check has can be generalized with the user-defined function given below.

```

1 CREATE OR REPLACE FUNCTION is_subset(
2     p_table_name text,
3     p_pk_name text,
4     p_col_a text,
5     p_col_b text
6 )
7 RETURNS boolean AS $$
8 DECLARE
9     count_a bigint;
10    count_unioned bigint;
11    v_query text;
12 BEGIN
13     -- Count of non-null values in column A
14     v_query := format('SELECT COUNT(%s) FROM %s WHERE %s IS NOT NULL',
15                       p_pk_name, p_table_name, p_col_a);
16     EXECUTE v_query INTO count_a;
17
18     -- Count after union of both columns
19     v_query := format('SELECT COUNT(*) FROM (
20                       (SELECT %s FROM %s WHERE %s IS NOT NULL)
21                       UNION
22                       (SELECT %s FROM %s WHERE %s IS NOT NULL)
23                       ) AS unioned',
24                       p_pk_name, p_table_name, p_col_a,
25                       p_pk_name, p_table_name, p_col_b);
26     EXECUTE v_query INTO count_unioned;
27
28     RETURN count_a = count_unioned;
29 END;
30 $$ LANGUAGE plpgsql;

```

Listing 2.11: Sample UDF for verifying if a column is a subset of another within the same table

- Finally, since the 'country', 'city', and 'currencycode' attributes depend more directly on the 'fullvisitorid' column, the all sessions table have been separated into two tables: session info and visitor info. The visitor info table collates the information that are directly dependent on the 'fullvisitorid'

2.2.2 The products, sales reports, and sales by sku datasets

- The data cleaning procedure is similar to the routine presented earlier for the all sessions table. The datatypes for the columns are selected based on the data and unavailable data are converted to NULLs,
- Using the `check_unique_values` user-defined function provided earlier, we can that confirm that the 'sku' column is a primary key (PK) for the products table, while the 'productsku' column is a PK for the sales report and sales by sku tables.
- However, the product names contained in the 'name' column are not necessarily unique because multiple SKUs can be the same product but with slight variations in color or size. Unfortunately, the product names do not seem to capture this variation.
- The product, sales report and sales by sku tables potentially contain similar and redundant information. From manual inspection and using the `is_subset` user-defined function, we can conclude that productsku information present in the sales report table is also present in the sales by sku table. This can be done via the following SQL query:

```
1  create temp table temp_combined_sales as
2  (select row_number() over() as unique_id,
3  sr.productsku, sr.total_ordered as sr_sold,
4  ss.productsku as ss_productsku, ss.total_ordered as ss_sold
5  from sales_report as sr
6  full outer join sales_by_sku as ss using(productsku));
7
8  select is_subset('temp_combined_sales', 'unique_id', 'ss_productsku',
9  'productsku');
```

Listing 2.12: Sample SQL query for inspecting duplicate attributes across tables

- This can also be checked by using a full outer join on the two tables and checking if there exists rows where `sales_report.productsku` is not null but `sales_by_sku.productsku` is null. Since no such row exists, this serves as a double confirmation. Hence, we can combine the sales by sku and sales report information into one table using a full join. By the same logic, productsku information in sales report and sales by sku is embedded in the products table. So we can left join the products table with the earlier created combined table.
- We can then derive two separate tables from this combined table: the product info and

sales info. The two tables are created from the combined table via the following SQL queries.

```
1  create table sales_info as (select p.sku as productsku, p.orderedquantity
2      ::int,s.quantitysold::int,
3      s.ratio::real
4  from products as p
5  left join temp_sales_report as s on p.sku = s.productsku);
6
7  create table product_info as (
8  select p.sku as productsku, p.name,p.stocklevel::int,
9  p.restockingleadtime::int,p.sentimentscore::real,
10 p.sentimentmagnitude::real
11 from products as p
12 left join temp_sales_report as s on p.sku = s.productsku);
```

Listing 2.13: Sample SQL query for creating the product info and sales info tables

- Here, clusters of information pertaining only to products have been separated from info pertaining to potential sales. Here, ‘orderedquantity’ represents potential refunds plus materialized sales.

2.2.3 The analytics dataset

- Again, the procedure for cleaning the analytics table is similar to that outlined for the previously discussed tables: unavailable data is processed as nulls, the appropriate data types are selected, and the price and revenue scales are corrected by dividing by 1×10^6 . The following SQL query transforms the data from the original analytics table and stores them into a newly created view.


```
1 create view vw_analytics as (  
2 select row_number() over() as unique_id,  
3        visitnumber::int,  
4        vistid as visitid,  
5        visitstarttime::int,  
6        date::date,  
7        fullvisitorid,  
8        userid,  
9        channelgrouping,  
10       socialengagementtype,  
11       unitssold::int,  
12       pageviews::int,  
13       timeonsite::int,  
14       bounces::int,  
15       (revenue::real)/1000000 as revenue,  
16       (unitprice::real)/1000000 as unitprice  
17 from analytics);  
18
```

Listing 2.14: Sample SQL query for cleaning the analytics table

- The data in the analytics table is a mess. This could be due to two main reasons. First, the revenue information provided is not unique to either the ‘visitid’ or the ‘fullvisitorid’, or even the composition of both columns. It is potentially missing a ‘productsku’ column to link the disaggregated revenue information to.
- A second reason is that it is difficult to understand the granularity of the data in the table. With logic similar to what was discussed in the all sessions dataset section, it can be determined that multiple unique ‘fullvisitorids’ can be associated to a single ‘visitid’. Also, a ‘fullvisitorid’ can be associated with different ‘visitids’. My guess is that a ‘visitid’ is built from the IP address which could be shared, for example by a full household, while the ‘fullvisitorid’ is built from more components like the IP address + stored cookies + browser/device information.
- Since the granularity can not be determined, we try to understand the revenue information from the aggregated view. On aggregating the revenue by ‘fullvisitorid’ and comparing it with the ‘totaltransactionsrevenue’ information in the all sessions table for the rows where there is a match in the ‘fullvisitorid’, we can ascertain that for the same values of

‘fullvisitorids’ in both the analytics and all sessions tables, the total transaction revenue is the same. The SQL query for this comparison is provided below.

```
1  select fullvisitorid, sum(revenue::numeric)
2  from analytics
3  where fullvisitorid in
4  (select fullvisitorid
5   from all_sessions
6   where totaltransactionrevenue is not null
7   order by fullvisitorid
8  )
9  group by fullvisitorid
10 order by fullvisitorid
```

Listing 2.15: Sample SQL query for inspecting duplicate attributes across tables

- However, the analytics table only contains 27 unique ‘fullvisitorids’ with country/city information, while the all sessions table contains 80. Hence, since all sessions table contains more information about the visitor, such as country and city, it is potentially more useful in relating the revenue to those attributes compared to the analytics table.

Chapter 3

Starting with questions

3.1 Question 1: Which cities and countries have the highest level of transaction revenues on the site?

Preliminaries

Information pertaining to the visitor info such as country and city is present in the original all sessions table. As discussed previously, this info is linked to ‘fullvisitorid‘ column. Additionally, revenue information is present in both the analytics and all sessions table, but only the ‘fullvisitorids‘ present in the all sessions table has information about the visitor. Finally, as the ‘transactionrevenue‘ column is a subset of the ‘totaltransactionrevenue‘ column, only the ‘totaltransactionrevenue‘ column in the all sessions table is relevant to answering this question.

SQL Queries

The country-specific query is provided in the listing below.

```
1 with cte_country_city_revenue as
2 (select v_info.country,v_info.city,se_info.trans_rev
3 from (select fullvisitorid, sum(totaltransactionrevenue) as trans_rev
4      from session_info
5      where totaltransactionrevenue is not null
6      group by fullvisitorid)as se_info
7 join visitor_info as v_info on se_info.fullvisitorid = v_info.fullvisitorid)
8
9
```

```

10 select c_rev.country, sum(c_rev.trans_rev) as transactionrevenue,
11 rank() over(order by sum(c_rev.trans_rev) desc) as rank
12 from cte_country_city_revenue as c_rev
13 where c_rev.country is not null
14 group by c_rev.country
15 order by rank;

```

Listing 3.1: Sample SQL query for ranking countries based on transaction revenue

The city-specific query is provided below.

```

1 with cte_country_city_revenue as
2 (select v_info.country,v_info.city,se_info.trans_rev
3 from (select fullvisitorid, sum(totaltransactionrevenue) as trans_rev
4 from session_info
5 where totaltransactionrevenue is not null
6 group by fullvisitorid)as se_info
7 join visitor_info as v_info on se_info.fullvisitorid = v_info.fullvisitorid)
8
9 select c_rev.city, sum(c_rev.trans_rev) as transactionrevenue,
10 rank() over(order by sum(c_rev.trans_rev) desc) as rank
11 from cte_country_city_revenue as c_rev
12 where c_rev.city is not null
13 group by c_rev.city
14 order by rank;

```

Listing 3.2: Sample SQL query for inspecting duplicate attributes across tables

Answer

Data Output

	country text	transactionrevenue double precision	rank bigint
1	United States	13166.359999999991	1
2	Israel	602	2
3	Australia	358	3
4	Canada	150.14999999999998	4
5	Switzerland	16.99	5

Figure 3.1: Countries with the highest level of transaction revenues on site.

Data Output

	city text	transactionrevenue double precision	rank bigint
1	San Francisco	1576.51	1
2	Sunnyvale	992.23	2
3	Atlanta	854.44	3
4	Palo Alto	608	4
5	Tel Aviv-Yafo	602	5
6	New York	598.35	6
7	Mountain View	483.36	7
8	Los Angeles	479.48	8
9	Chicago	449.52	9
10	Sydney	358	10
11	Seattle	358	10
12	San Jose	262.38	12
13	Austin	157.78	13
14	Nashville	157	14

Figure 3.2: Cities with the highest level of transaction revenues on site.

3.2 Question 2: What is the average number of products ordered from visitors in each city and country?

SQL Queries

The country-specific query is provided in the listing below.

```

1 with cte_country_city_ordered as
2 (select *
3  from (select si.fullvisitorid,vi.country,vi.city,si.productsku
4         from session_info as si
5         join visitor_info as vi using(fullvisitorid))as se_info
6  join sales_info as s_info on se_info.productsku = s_info.productsku)
7
8 select c_rev.country, avg(c_rev.orderedquantity) as avg_ordered,
9 rank() over(order by avg(c_rev.orderedquantity) desc) as rank
10 from cte_country_city_ordered as c_rev
11 where c_rev.country is not null
12 group by c_rev.country

```

```
13 order by rank;
```

Listing 3.3: Sample SQL query for ranking countries based on transaction revenue

The city-specific query is provided below.

```
1 with cte_country_city_ordered as
2 (select *
3  from (select si.fullvisitorid,vi.country,vi.city,si.productsku
4         from session_info as si
5         join visitor_info as vi using(fullvisitorid))as se_info
6  join sales_info as s_info on se_info.productsku = s_info.productsku)
7
8  select c_rev.city, avg(c_rev.orderedquantity) as avg_ordered,
9  rank() over(order by avg(c_rev.orderedquantity) desc) as rank
10 from cte_country_city_ordered as c_rev
11 where c_rev.city is not null
12 group by c_rev.city
13 order by rank;
```

Listing 3.4: Sample SQL query for inspecting duplicate attributes across tables

3.2. Question 2: What is the average number of products ordered from visitors in each city
Chapter 3. Starting with questions and country?

Answer

Data Output

	country text	avg_ordered numeric	rank bigint
1	Mali	3786.0000000000000000	1
2	Montenegro	3786.0000000000000000	1
3	Papua New Guinea	2558.0000000000000000	3
4	Réunion	2538.0000000000000000	4
5	Georgia	2506.4000000000000000	5
6	Côte d'Ivoire	1928.5000000000000000	6
7	Moldova	1893.0000000000000000	7
8	Tanzania	1429.0000000000000000	8
9	Trinidad & Tobago	1379.5000000000000000	9
10	Armenia	1351.0000000000000000	10
11	Oman	1330.0000000000000000	11
12	Ethiopia	1239.0000000000000000	12
13	Bulgaria	1231.72727272727273	13
14	Portugal	1219.8571428571428571	14
15	Morocco	1203.4285714285714286	15
16	Sint Maarten	1100.0000000000000000	16

total rows: 133 of 133 Query complete 00:00:00.098

Figure 3.3: Countries with the highest level of average number of products ordered on site.

Data Output			
	city text	avg_ordered numeric	rank bigint
1	Council Bluffs	7589.0000000000000000	1
2	Bellflower	3786.0000000000000000	2
3	Cork	3786.0000000000000000	2
4	Santiago	3607.0000000000000000	4
5	Bellingham	2836.0000000000000000	5
6	Detroit	2748.0000000000000000	6
7	Westville	2299.0000000000000000	7
8	Santa Fe	1932.5000000000000000	8
9	Nashville	1886.0000000000000000	9
10	Brno	1548.0000000000000000	10
11	Zhongli District	1492.0000000000000000	11
12	Ghent	1351.0000000000000000	12
13	Gurgaon	1335.8571428571428571	13
14	Rome	1310.4000000000000000	14
15	Mississauga	1305.2500000000000000	15
16	Moscow	1291.8000000000000000	16

Total rows: 250 of 250 Query complete 00:00:00.073

Figure 3.4: Cities with the highest level of average number of products ordered on site.

3.3 Question 3: Is there any pattern in the types (product categories) of products ordered from visitors in each city and country?

SQL Queries

The country-specific query is provided in the listing below.

```

1 with cte_country_city_category as
2 (select *
3  from (select si.fullvisitorid,vi.country,vi.city,si.v2productcategory
4         from session_info as si
5         join visitor_info as vi using(fullvisitorid))as se_info
6 )
7

```



```
8 select country, v2productcategory, freq
9 from (select c_rev.country, c_rev.v2productcategory, count(*) as freq,
10 rank() over(partition by c_rev.country order by count(*) desc) as rank
11 from cte_country_city_category as c_rev
12 where c_rev.country is not null
13 group by c_rev.country, c_rev.v2productcategory
14 order by freq desc)
15 where rank = 1;
```

Listing 3.5: Sample SQL query for ranking countries based on transaction revenue

The city-specific query is provided below.


```
1 with cte_country_city_category as
2 (select *
3 from (select si.fullvisitorid, vi.country, vi.city, si.v2productcategory
4        from session_info as si
5        join visitor_info as vi using(fullvisitorid)) as se_info
6 )
7
8 select city, v2productcategory, freq
9 from (select c_rev.city, c_rev.v2productcategory, count(*) as freq,
10 rank() over(partition by c_rev.city order by count(*) desc) as rank
11 from cte_country_city_category as c_rev
12 where c_rev.city is not null
13 group by c_rev.city, c_rev.v2productcategory
14 order by freq desc)
15 where rank = 1;
```


Listing 3.6: Sample SQL query for inspecting duplicate attributes across tables

Answer

It is evident from the figure, that t-shirts for men and Youtube branded merchandise are extremely popular across countries and cities.

3.3. Question 3: Is there any pattern in the types (product categories) of products ordered from visitors in each city and country? Chapter 3. Starting with questions

Data Output 



	country text	v2productcategory text	freq bigint
1	United States	Home/Apparel/Men's/Men's-T-Shirts/	936
2	United Kingdom	Home/Shop by Brand/YouTube/	255
3	India	Home/Shop by Brand/YouTube/	219
4	Germany	Home/Shop by Brand/YouTube/	128
5	Canada	Home/Shop by Brand/YouTube/	118
6	Australia	Home/Shop by Brand/YouTube/	71
7	Japan	Home/Apparel/Men's/Men's-T-Shirts/	53
8	France	Home/Shop by Brand/YouTube/	43
9	Netherlands	Home/Shop by Brand/YouTube/	37
10	Italy	Home/Shop by Brand/YouTube/	31
11	Czechia	Home/Shop by Brand/YouTube/	30
12	Brazil	Home/Apparel/Men's/Men's-T-Shirts/	28
13	Russia	Home/Shop by Brand/YouTube/	28
14	Indonesia	Home/Shop by Brand/YouTube/	28
15	Singapore	Home/Apparel/Men's/Men's-T-Shirts/	23
16	Mexico	Home/Shop by Brand/YouTube/	23

Figure 3.5: Countries together with the most popular product category on site.

Data Output			
	city text	v2productcategory text	freq bigint
1	Mountain View	Home/Apparel/Men's/Men's-T-Shirts/	110
2	New York	Home/Apparel/Men's/Men's-T-Shirts/	77
3	London	Home/Shop by Brand/YouTube/	62
4	San Francisco	Home/Apparel/Men's/Men's-T-Shirts/	47
5	Sunnyvale	Home/Apparel/Men's/Men's-T-Shirts/	38
6	Chennai	Home/Shop by Brand/YouTube/	37
7	Los Angeles	Home/Apparel/Men's/Men's-T-Shirts/	33
8	San Jose	Home/Apparel/Men's/Men's-T-Shirts/	29
9	Palo Alto	Home/Nest/Nest-USA/	26
10	Chicago	Home/Apparel/Men's/Men's-T-Shirts/	23
11	Bengaluru	Home/Shop by Brand/YouTube/	20
12	Melbourne	Home/Shop by Brand/YouTube/	19
13	Toronto	Home/Apparel/Men's/Men's-T-Shirts/	18
14	Sydney	Home/Shop by Brand/YouTube/	18
15	New Delhi	Home/Apparel/Men's/Men's-T-Shirts/	17
16	Dublin	Home/Shop by Brand/YouTube/	16

1 rows: 412 of 412 Query complete 00:00:00.069

Figure 3.6: Cities together with the most popular product category on site.

3.4 Question 4: What is the top-selling product from each city/country? Can we find any pattern worthy of noting in the products sold?

SQL Queries

The country-specific query is provided in the listing below.

```

1 with cte_country_city_category as
2 (select se_info.country, se_info.city, se_info.productsku,
3    s_info.quantitiesold, p_info.name
4 from (select si.fullvisitorid, vi.country, vi.city, si.productsku
5    from session_info as si
6    join visitor_info as vi using(fullvisitorid)) as se_info
7 join sales_info as s_info on se_info.productsku = s_info.productsku

```

```
8 join product_info as p_info on s_info.productsku = p_info.productsku
9 )
10
11
12 select country, name, freq
13 from (select c_rev.country, c_rev.name, count(*) as freq,
14 rank() over(partition by c_rev.country order by count(*) desc) as rank
15 from cte_country_city_category as c_rev
16 where c_rev.country is not null
17 group by c_rev.country, c_rev.name
18 order by freq desc)
19 where rank = 1;
```

Listing 3.7: Sample SQL query for ranking countries based on transaction revenue

The city-specific query is provided below.

```
1 with cte_country_city_category as
2 (select se_info.country, se_info.city, se_info.productsku,
3 s_info.quantitysold, p_info.name
4 from (select si.fullvisitorid, vi.country, vi.city, si.productsku
5 from session_info as si
6 join visitor_info as vi using(fullvisitorid)) as se_info
7 join sales_info as s_info on se_info.productsku = s_info.productsku
8 join product_info as p_info on s_info.productsku = p_info.productsku
9 )
10
11 select city, name, freq
12 from (select c_rev.city, c_rev.name, count(*) as freq,
13 rank() over(partition by c_rev.city order by count(*) desc) as rank
14 from cte_country_city_category as c_rev
15 where c_rev.city is not null
16 group by c_rev.city, c_rev.name
17 order by freq desc)
18 where rank = 1;
```

Listing 3.8: Sample SQL query for inspecting duplicate attributes across tables

3.4. Question 4: What is the top-selling product from each city/country? Can we find any pattern worthy of noting in the products sold?

Answer

Data Output			
	country text	name text	freq bigint
1	United States	Men's 100% Cotton Short Sleeve Hero Tee White	170
2	United Kingdom	Twill Cap	23
3	India	Custom Decals	22
4	Canada	Twill Cap	16
5	Germany	Twill Cap	14
6	Australia	Twill Cap	11
7	France	Men's 100% Cotton Short Sleeve Hero Tee White	8
8	Japan	Men's Vintage Tank	7
9	Netherlands	Twill Cap	7
10	Japan	Men's 100% Cotton Short Sleeve Hero Tee White	7
11	Brazil	Men's Vintage Badge Tee Black	6
12	Mexico	Men's Short Sleeve Hero Tee Black	6
13	Czechia	Hard Cover Journal	6
14	Indonesia	22 oz Bottle Infuser	6
15	Brazil	Trucker Hat	6
16	Taiwan	Men's 100% Cotton Short Sleeve Hero Tee White	6
Total rows: 380 of 380 Query complete 00:00:00.069			

Figure 3.7: Countries together with the most popular product sold on site.

3.4. Question 4: What is the top-selling product from each city/country? Can we find any pattern worthy of noting in the products sold? Chapter 3. Starting with questions

Data Output			
	city text	name text	freq bigint
1	Mountain View	Cam Indoor Security Camera - USA	24
2	New York	Twill Cap	15
3	San Francisco	Men's 100% Cotton Short Sleeve Hero Tee White	12
4	London	Twill Cap	10
5	San Jose	Men's 100% Cotton Short Sleeve Hero Tee White	8
6	Chicago	Men's 100% Cotton Short Sleeve Hero Tee White	7
7	Palo Alto	Cam Outdoor Security Camera - USA	7
8	Los Angeles	Men's 100% Cotton Short Sleeve Hero Tee White	7
9	Sunnyvale	Cam Indoor Security Camera - USA	6
10	Sunnyvale	Men's Short Sleeve Hero Tee Charcoal	6
11	Sunnyvale	Twill Cap	6
12	Sunnyvale	Cam Outdoor Security Camera - USA	6
13	Sydney	Trucker Hat	5
14	Chennai	22 oz Bottle Infuser	5
15	Toronto	Trucker Hat	5
16	Hyderabad	Men's 100% Cotton Short Sleeve Hero Tee White	5

total rows: 743 of 743 Query complete 00:00:00.153

Figure 3.8: Cities together with the most popular product sold on site.

Chapter 4

ERD for database

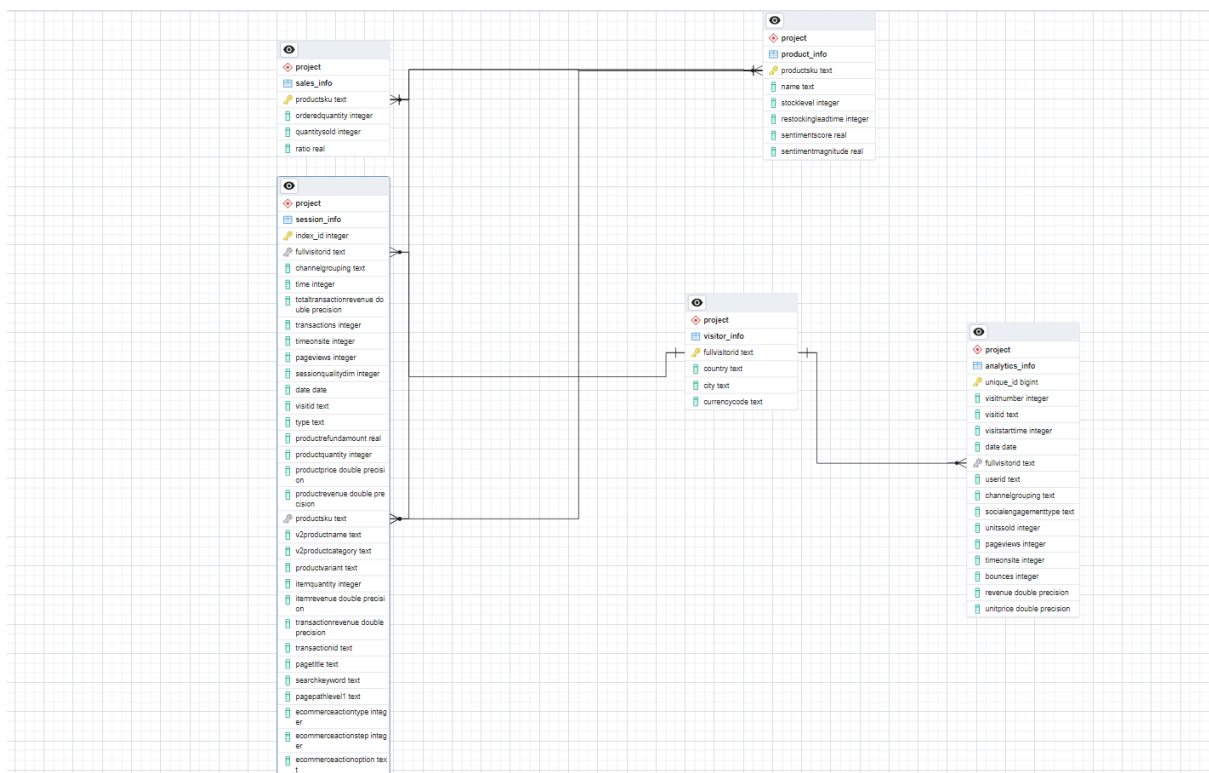


Figure 4.1: ERD for database

Chapter 5

Starting with data

5.1 Question 1: What products are among the top 10 sold but whose stock level are currently below the average stock level for all products?

SQL Queries

The query is provided in the listing below.

```
1 select productsku, name, stocklevel
2 from project.product_info
3 where productsku in (select productsku
4                       from project.sales_info
5                       order by quantitysold desc
6                       limit 10)
7 and stocklevel < (select avg(stocklevel)
8                  from project.product_info)
9 order by stocklevel
```

Listing 5.1: Sample SQL query for ranking countries based on transaction revenue

Answer

It is evident from the figure, that t-shirts for men and Youtube branded merchandise are extremely popular across countries and cities.

5.1. Question 1: What products are among the top 10 sold but whose stock level are currently
Chapter 5. Starting with data below the average stock level for all products?

Data Output			
	country text	name text	freq bigint
1	United States	Men's 100% Cotton Short Sleeve Hero Tee White	170
2	United Kingdom	Twill Cap	23
3	India	Custom Decals	22
4	Canada	Twill Cap	16
5	Germany	Twill Cap	14
6	Australia	Twill Cap	11
7	France	Men's 100% Cotton Short Sleeve Hero Tee White	8
8	Japan	Men's Vintage Tank	7
9	Netherlands	Twill Cap	7
10	Japan	Men's 100% Cotton Short Sleeve Hero Tee White	7
11	Brazil	Men's Vintage Badge Tee Black	6
12	Mexico	Men's Short Sleeve Hero Tee Black	6
13	Czechia	Hard Cover Journal	6
14	Indonesia	22 oz Bottle Infuser	6
15	Brazil	Trucker Hat	6
16	Taiwan	Men's 100% Cotton Short Sleeve Hero Tee White	6
Total rows: 380 of 380		Query complete 00:00:00.069	

Figure 5.1: Countries together with the most popular product sold on site.

5.1. Question 1: What products are among the top 10 sold but whose stock level are currently below the average stock level for all products? Chapter 5. Starting with data

Data Output				
	city text	name text	freq bigint	
1	Mountain View	Cam Indoor Security Camera - USA	24	
2	New York	Twill Cap	15	
3	San Francisco	Men's 100% Cotton Short Sleeve Hero Tee White	12	
4	London	Twill Cap	10	
5	San Jose	Men's 100% Cotton Short Sleeve Hero Tee White	8	
6	Chicago	Men's 100% Cotton Short Sleeve Hero Tee White	7	
7	Palo Alto	Cam Outdoor Security Camera - USA	7	
8	Los Angeles	Men's 100% Cotton Short Sleeve Hero Tee White	7	
9	Sunnyvale	Cam Indoor Security Camera - USA	6	
10	Sunnyvale	Men's Short Sleeve Hero Tee Charcoal	6	
11	Sunnyvale	Twill Cap	6	
12	Sunnyvale	Cam Outdoor Security Camera - USA	6	
13	Sydney	Trucker Hat	5	
14	Chennai	22 oz Bottle Infuser	5	
15	Toronto	Trucker Hat	5	
16	Hyderabad	Men's 100% Cotton Short Sleeve Hero Tee White	5	
Total rows: 743 of 743		Query complete 00:00:00.153		

Figure 5.2: Cities together with the most popular product sold on site.