BUAN 6320.003 – Database Foundations for Business Analytics

Project 2

PROJECT GROUP – 10

SUPER STORE

Members

Sai Krishna Adabala

Sai Madhan Muthyam

Ashritha Kotla

Govardhan Yadav Pomkom

Sahoo Swarendini

Gutta Gowtham Vamsi

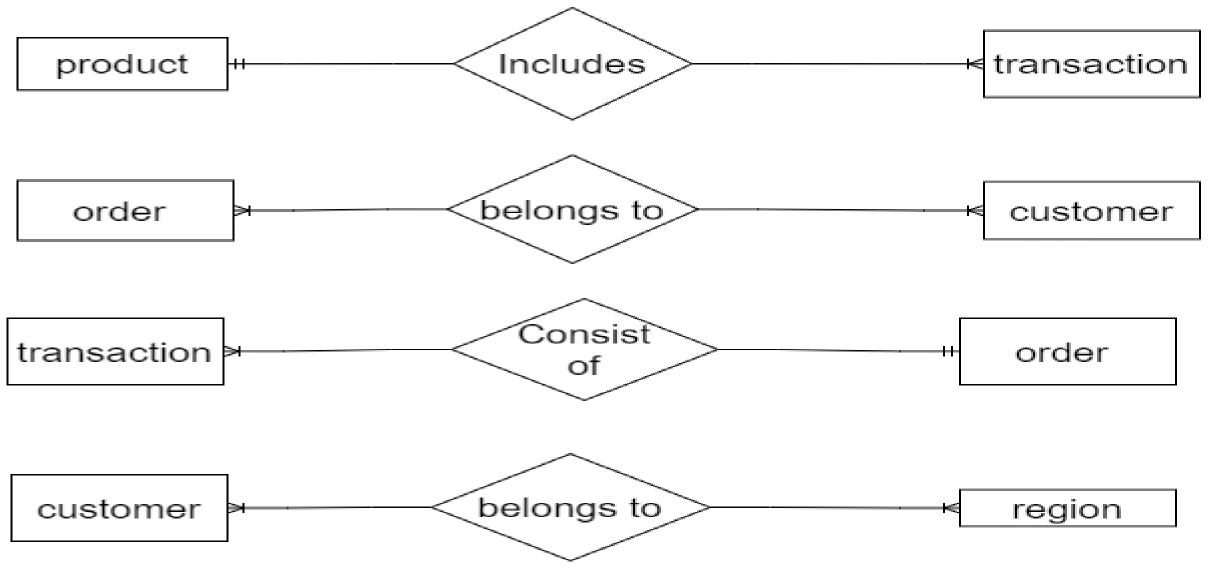
Schema Design:

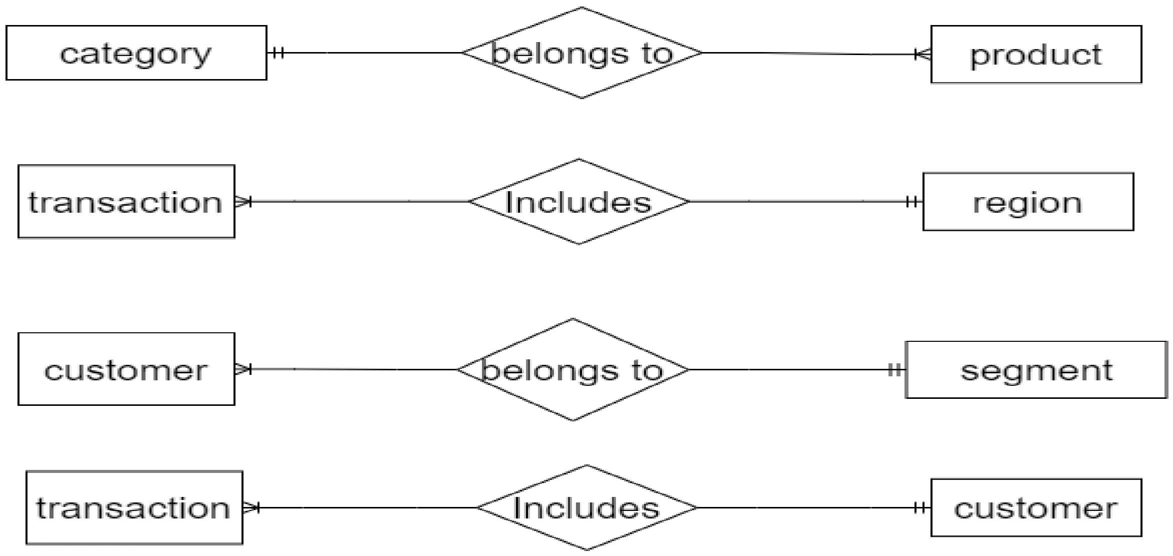
a) **Entities, their attributes, their primary keys, and relationships**

A total of seven entities are defined and details of their attributes and primary keys are mentioned in the below table.

Entity	Attributes
transaction	**unique_id (PK)**
	sales
	quantity
	discount
	profit
	shipping_cost
order	**order_id (PK)**
	order_date (PK)
customer	**customer_id (PK)**
	customer_name
segment	**segment_id (PK)**
	segment_name
region	**region_id (PK)**
	region_name
product_id	**product_id (PK)**
	product_name
category	**category_id (PK)**
	category_name

Relationships between entities are depicted below.





b) Constraints:

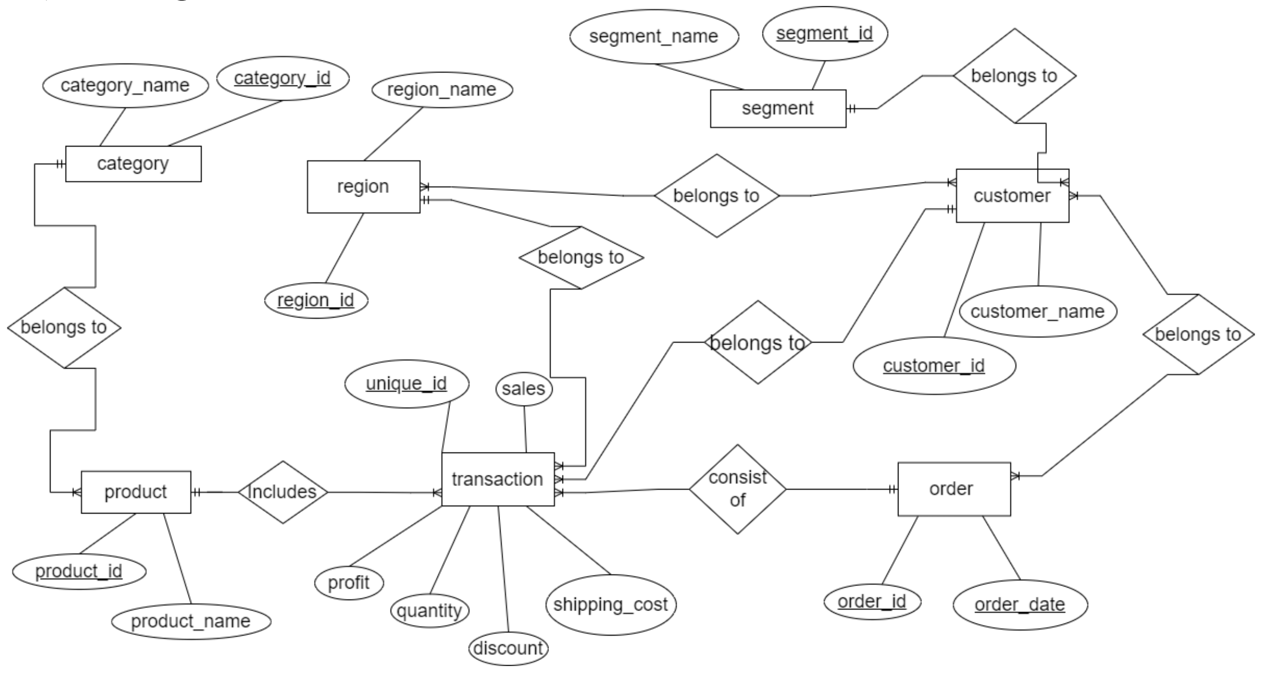
Based up on the data and relations between the attribute following constraints are defined

- Product – Transaction (One-to-Many relationship)
  Each Transaction is associated with exactly one product, whereas each product is ordered at least once.
- Order – Customer (Many-to-Many relationship)
  It is observed that each order_id and order_date is associated with one or more customer (which is unusual but this dataset has), whereas a customer had placed one or more orders.
- Order – Transaction (One-to-Many relationship)
  Each Transaction is associated with exactly one order, whereas an order can include one or more products and accordingly one or more transactions.
- Customer – Transaction (One-to-Many relationship)
  Each Transaction is associated with exactly one customer, whereas an customer can include one or more products and accordingly one or more transactions.
- Region – Customer (Many-to-Many relationship)
  Each customer has business in multiple regions (one or more) and also each region has multiple customers (one or more).
- Product – Category (One-to-Many relationship)
  Each Product is associated with exactly one Category, whereas each Category has one or more products.
- Region – Transaction (One-to-Many relationship)
  Each Transaction is associated with exactly one region, whereas for a given region there are one or more transaction.
- Customer – Segment (One-to-Many relationship)
  Each Customer is belongs to one segment only, whereas each segment has one or more customers.

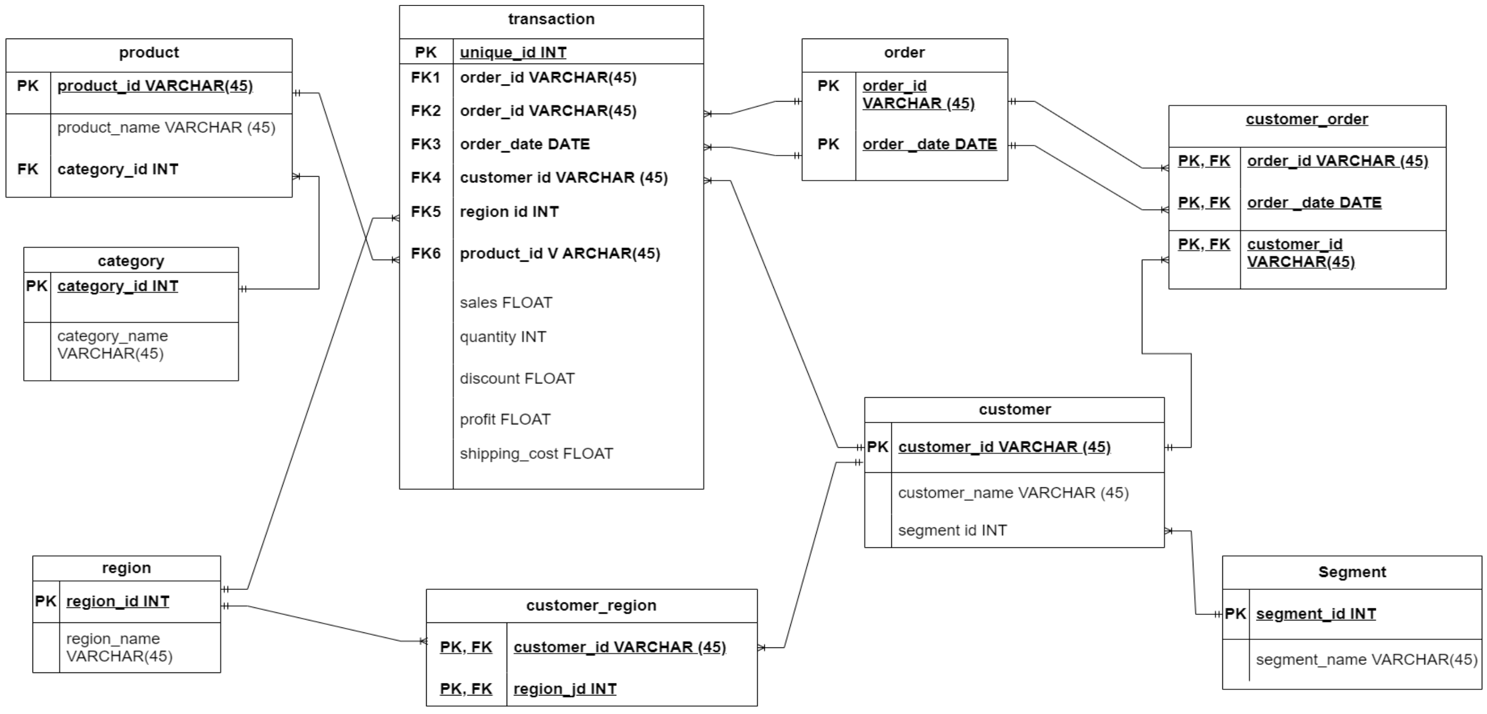
# since this schema is designed based on the existing records, so each product or customer is a part of transaction at least once, so minimum cardinality for all the relations is 1.

Domain Constraints: Details of type of data for a given attribute are provided in project_1 report, details of same are provided in Appendix.

c) ER diagram:



d) Translation of ER diagram in to relations:



product	
PK	product_id VARCHAR(45)
	product_name VARCHAR (45)
FK	category_id INT

category	
PK	category_id INT
	category_name VARCHAR(45)

transaction	
PK	unique_id INT
FK1	order_id VARCHAR(45)
FK2	order_id VARCHAR(45)
FK3	order_date DATE
FK4	customer id VARCHAR (45)
FK5	region id INT
FK6	product_id V ARCHAR(45)
	sales FLOAT
	quantity INT
	discount FLOAT
	profit FLOAT
	shipping_cost FLOAT

order	
PK	order_id VARCHAR (45)
PK	order _date DATE

customer_order	
PK, FK	order_id VARCHAR (45)
PK, FK	order _date DATE
PK, FK	customer_id VARCHAR(45)

customer	
PK	customer_id VARCHAR (45)
	customer_name VARCHAR (45)
	segment id INT

region	
PK	region_id INT
	region_name VARCHAR(45)

customer_region	
PK, FK	customer_id VARCHAR (45)
PK, FK	region_jd INT

Segment	
PK	segment_id INT
	segment_name VARCHAR(45)

Schema Normalization

a) **Functional dependencies from schema**

Following are all the functional dependencies from schema

- {unique_id} -> {order_id, order_date}
- {unique_id} -> {customer_id}
- {unique_id} -> {sales, quantity, discount, profit, shipping_cost}
- {product_id} -> {product_name, category_id}
- {category_id} -> {category_name}
- {customer_id} -> {customer_name, segment_id}
- {unique_id} -> {segment_name, region_name, category_name}
- {unique_id} -> {product_id}
- {unique_id} -> {region_id}
- {region_id} -> {region_name}
- {segment_id} -> {segment_name}

b) **Verification of minimal keys**

- <u>transaction table:</u>
  Since the transaction table consists of only one primary key {unique_id} and it functionally determines all other attributes in the table, there is no further evaluation of minimal primary key is required.
- <u>order table:</u>
  In this table, primary key is a combination of {order_id, order_date} is minimal because no subset of these attributes can functionally determine all other attributes in the table, this is an example of composite primary key.
- <u>category table:</u>
  Since the category table consists of only one primary key {category_id} and it functionally determines other attributes in the table, there is no further evaluation of minimal primary key is required.
- <u>product table:</u>
  Since the product table consists of only one primary key {product_id} and it functionally determines other attributes in the table, there is no further evaluation of minimal primary key is required.
- <u>segment table:</u>
  Since the segment table consists of only one primary key {segment _id} and it functionally determines other attributes in the table, there is no further evaluation of minimal primary key is required.

- region table:
  Since the region table consists of only one primary key {region _id} and it functionally determines other attributes in the table, there is no further evaluation of minimal primary key is required.
- customer table:
  Since the customer table consists of only one primary key {customer_id} and it functionally determines all other attributes in the table, there is no further evaluation of minimal primary key is required.
- customer_region table:
  Since customer & region entities are involved in the many-to-many relationship, another relation is created "customer_region" to represent the many-to-many relationship itself.
  This new relation has foreign keys, corresponding to the primary keys of the two relations representing the two entities involved in the many-to-many relationship.
  The two foreign keys form the composite primary key of the new relation and hence {customer_id and region_id} is a composite primary key for customer_region table.
  In customer_region table primary key {customer_id and region_id} is minimal because no subset of these attributes can functionally determine all other attributes in the table.
- order_ customer table:
  Since customer & order entities are involved in the many-to-many relationship, another relation is created "order_ customer" to represent the many-to-many relationship itself.
  Similar to customer_region table, customer_order table has composite primary key which is {customer_id, order_id, order_date}.
  {customer_id, order_id, order_date} is minimal because no subset of these attributes can functionally determine all other attributes in the table.

c) Verification whether schema is in BCNF {Boyce-Codd Normal Form}

To verify whether the current schema is in BCNF, all the functional dependencies that are mentioned above need to be verified, details of the same are mentioned below.

- {unique_id} -> {order_id, order_date}
  unique_id, order_id, order_date are in transaction table, the determinant unique_id is a key. **– Valid**

- {unique_id} -> {customer_id}
  unique_id, customer_id are in transaction table, the determinant unique_id is a key. **– Valid**

- {unique_id} -> {sales, quantity, discount, profit, shipping_cost}
  unique_id, sales, quantity, discount, profit, shipping_cost are in transaction table, the determinant unique_id is a key. **– Valid**

- {unique_id} -> {product_id}
  unique_id, product_id are in transaction table, the determinant unique_id is a key. **– Valid**

- {unique_id} -> {region_id}
  unique_id, region_id are in transaction table, the determinant unique_id is a key. **– Valid**
- {category _id} -> {category_name}
  category_id, category_name are in category table, the determinant { category_id } is a key. **– Valid**
- {product_id} -> {product_name, category_id}
  product_id, product_name, category_id are in product table, the determinant {product_id} is a key. **– Valid**
- {segment_id} -> {segment_name}
  segment_id, segment_name are in region table, the determinant {segment_id} is a key. **– Valid**
- {region_id} -> {region_name}
  region_id, region_name are in region table, the determinant {region_id} is a key. **– Valid**
- {customer_id} -> {customer_name, segment_id}
  customer_id, customer_name and segment are in customer table, the determinant {customer_id} is a key. **– Valid**

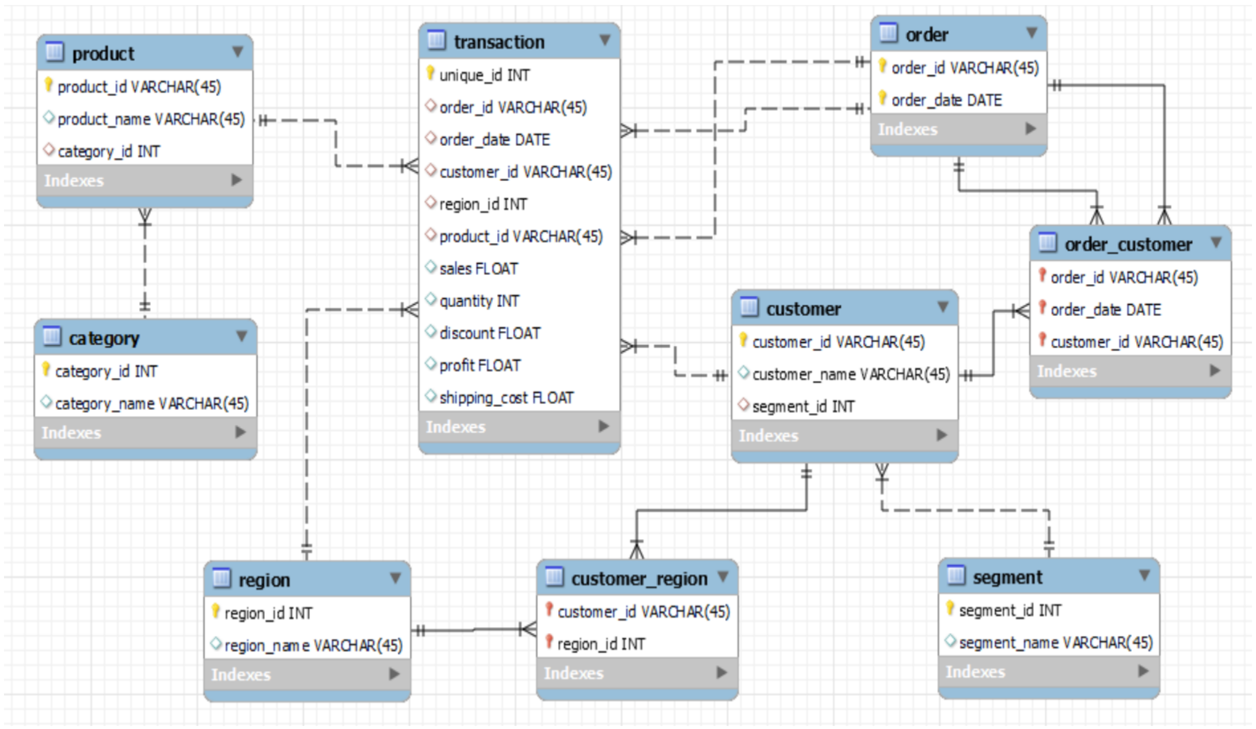
- {unique_id} -> {segment_name, region_name, category_name}
  Since all of them are not in same table, **above functional dependency is valid**.

  Since all the functional dependencies are valid, the designed schema is in BCNF (Boyce-Codd Normal Form).

  d) & e) Since there is no violation of BCNF, no further decomposition and updation of Schema is required.

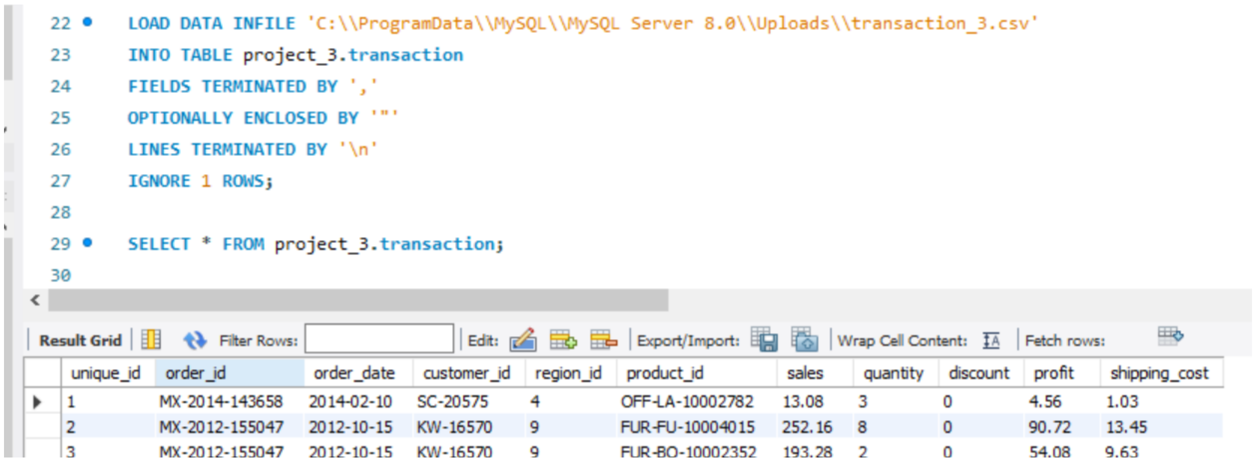
Schema (snapshot)

After creating tables and defining foreign key constraints, using *Reverse Engineer* following schema is developed.



Data Import:

Data for transaction table was successfully imported using the following query.



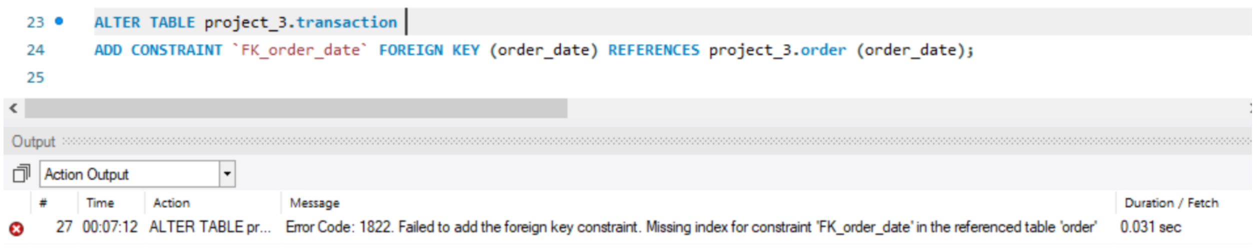
```
LOAD DATA INFILE 'C:\\ProgramData\\MySQL\\MySQL Server 8.0\\Uploads\\transaction_3.csv'
INTO TABLE project_3.transaction
FIELDS TERMINATED BY ','
OPTIONALLY ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 ROWS;

SELECT * FROM project_3.transaction;
```

unique_id	order_id	order_date	customer_id	region_id	product_id	sales	quantity	discount	profit	shipping_cost
1	MX-2014-143658	2014-02-10	SC-20575	4	OFF-LA-10002782	13.08	3	0	4.56	1.03
2	MX-2012-155047	2012-10-15	KW-16570	9	FUR-FU-10004015	252.16	8	0	90.72	13.45
3	MX-2012-155047	2012-10-15	KW-16570	9	FUR-BO-10002352	193.28	2	0	54.08	9.63

Similarly, data for other tables are successfully imported and SQL queries which are used for importing data into the other tables are provided in the Appendix.

Challenges incurred while importing the data:

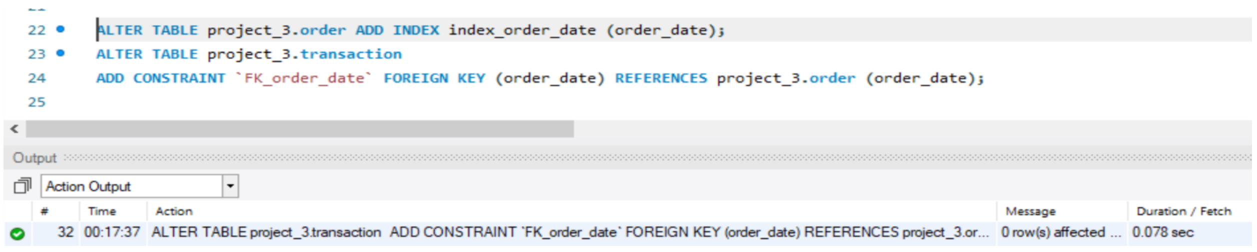
1. When adding a foreign key constraint for order_date column in transaction table, reference for the foreign key constraint is order table and order_date column, received the following error:
   Error Code: 1822. Failed to add the foreign key constraint. Missing index for constraint 'FK_order_date' in the referenced table 'order'.

   ```
   ALTER TABLE project_3.transaction
   ADD CONSTRAINT `FK_order_date` FOREIGN KEY (order_date) REFERENCES project_3.order (order_date);
   ```

   ```
   Output
   Action Output
   #   Time      Action           Message                                                                                                                  Duration / Fetch
   27  00:07:12  ALTER TABLE pr...  Error Code: 1822. Failed to add the foreign key constraint. Missing index for constraint 'FK_order_date' in the referenced table 'order'   0.031 sec
   ```

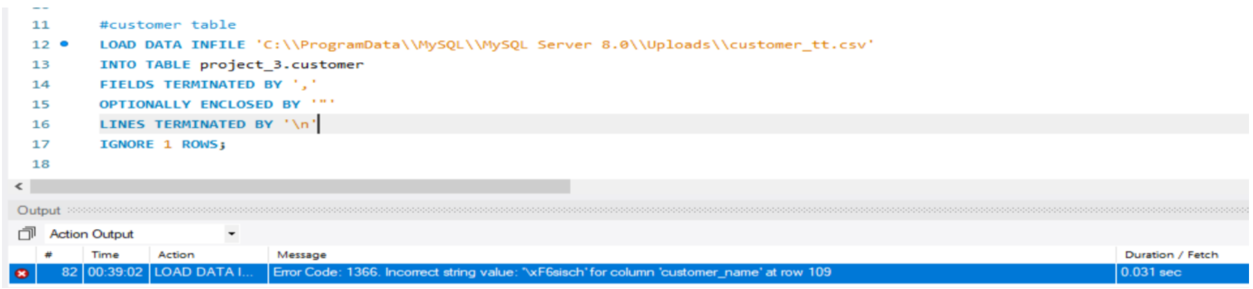
   Resolution:
   After adding index to the order_date in the order table, executed the same code and successfully added foreign key constraint.

   ```
   ALTER TABLE project_3.order ADD INDEX index_order_date (order_date);
   ALTER TABLE project_3.transaction
   ADD CONSTRAINT `FK_order_date` FOREIGN KEY (order_date) REFERENCES project_3.order (order_date);
   ```

   ```
   Output
   Action Output
   #   Time      Action                                                                                                      Message                Duration / Fetch
   32  00:17:37  ALTER TABLE project_3.transaction  ADD CONSTRAINT `FK_order_date` FOREIGN KEY (order_date) REFERENCES project_3.or...  0 row(s) affected ...  0.078 sec
   ```

2. When importing data into customer table, following error was encountered:
   Error Code: 1366. Incorrect string value: '\xF6sisch' for column 'customer_name' at row 109.

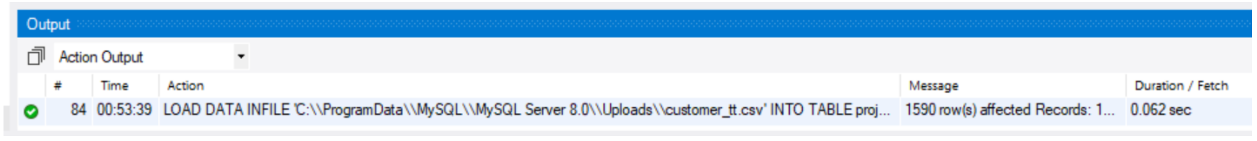
   

   ```
   #customer table
   LOAD DATA INFILE 'C:\\ProgramData\\MySQL\\MySQL Server 8.0\\Uploads\\customer_tt.csv'
   INTO TABLE project_3.customer
   FIELDS TERMINATED BY ','
   OPTIONALLY ENCLOSED BY '"'
   LINES TERMINATED BY '\n'
   IGNORE 1 ROWS;
   ```

   ```
   Output
   Action Output
   #   Time      Action         Message                                                                          Duration / Fetch
   82  00:39:02  LOAD DATA I...  Error Code: 1366. Incorrect string value: '\xF6sisch' for column 'customer_name' at row 109  0.031 sec
   ```

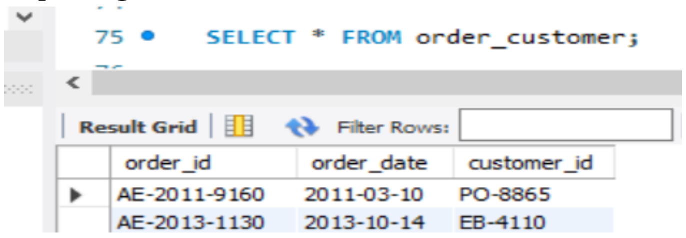
   Resolution:
   After analyzing the data in CSV file corresponding to customer table, at row number 109, customer name included a character ‘***ü***’, the same is replaced with an alphabet ‘***u***’ and tried to load the table, similar errors were incurred while loading the table due to special

character like **'ä'** and **'ö'** accordingly all these characters are replaced with alphabet **'a'** and **'o'** respectively and successfully loaded the customer table.



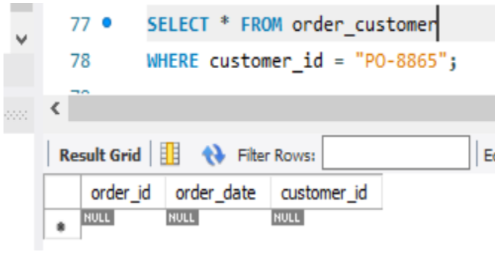
```
Output
Action Output
#   Time      Action                                                                                                    Message                                    Duration / Fetch
84  00:53:39  LOAD DATA INFILE 'C:\\ProgramData\\MySQL\\MySQL Server 8.0\\Uploads\\customer_tt.csv' INTO TABLE proj...  1590 row(s) affected Records: 1...   0.062 sec
```

3. Possibility of improper encoding of data in customer_id column (last column) in the order_customer table. One of the weirdest errors that were encountered, after importing the data successfully while selecting all the records in the table, we are able to get all the records in the output, as mentioned in the below screenshot, but while selecting a record which has a *customer_id = "PO-8865"*, null values were the output whereas it can be observed below that there is record corresponding to *customer_id = "PO-8865",* whereas when tried to select a record with *order_id = "AE-2011-9160"* which corresponds to *customer_id = "PO-8865"*, the output of the code is as expected, so possible reason for this error might be improper encoding of data in customer_id column (last column) in the order table during importing of data.



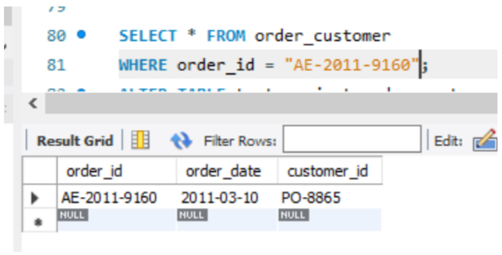
```
SELECT * FROM order_customer;
```

order_id	order_date	customer_id
AE-2011-9160	2011-03-10	PO-8865
AE-2013-1130	2013-10-14	EB-4110



```
SELECT * FROM order_customer
WHERE customer_id = "PO-8865";
```

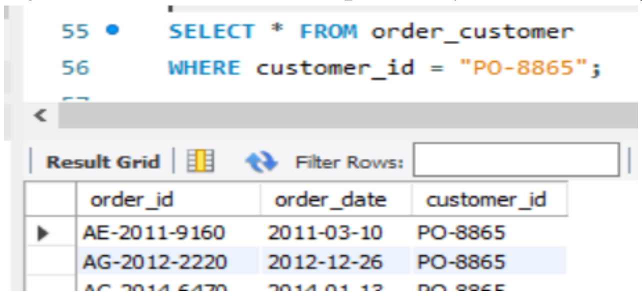
order_id	order_date	customer_id
NULL	NULL	NULL



```
SELECT * FROM order_customer
WHERE order_id = "AE-2011-9160";
```

order_id	order_date	customer_id
AE-2011-9160	2011-03-10	PO-8865
NULL	NULL	NULL

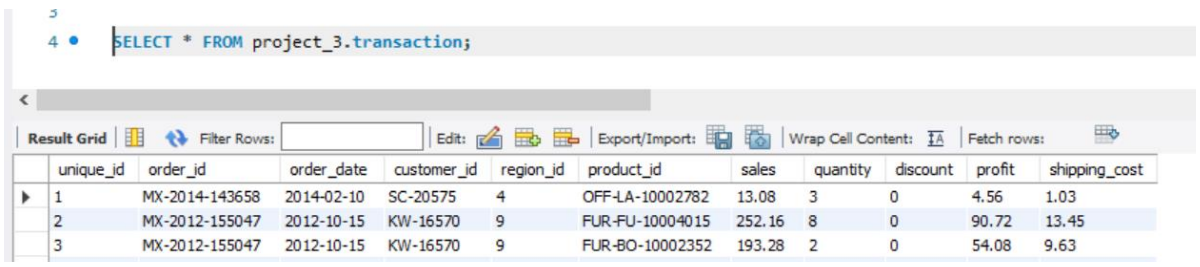
Resolution:
Since this error is observed in the last column of the table, after dropping the table, same table is created again with an additional column and accordingly data is imported, later the additional column is dropped and ran the similar queries with customer_id in WHERE clause and successfully obtained all the orders placed by the customer_id = "PO-8865".



```
SELECT * FROM order_customer
WHERE customer_id = "PO-8865";
```

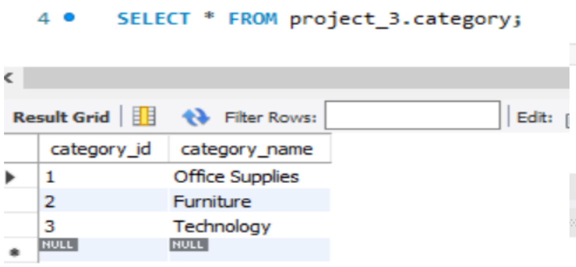
order_id	order_date	customer_id
AE-2011-9160	2011-03-10	PO-8865
AG-2012-2220	2012-12-26	PO-8865

Data Cleaning and Database Testing: From the below screenshot, it can be inferred that all tables have desired columns, and they have desired values.



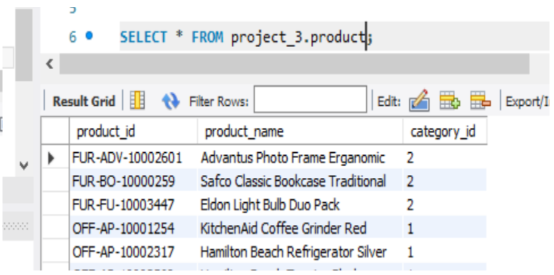
```sql
SELECT * FROM project_3.transaction;
```

unique_id	order_id	order_date	customer_id	region_id	product_id	sales	quantity	discount	profit	shipping_cost
1	MX-2014-143658	2014-02-10	SC-20575	4	OFF-LA-10002782	13.08	3	0	4.56	1.03
2	MX-2012-155047	2012-10-15	KW-16570	9	FUR-FU-10004015	252.16	8	0	90.72	13.45
3	MX-2012-155047	2012-10-15	KW-16570	9	FUR-BO-10002352	193.28	2	0	54.08	9.63



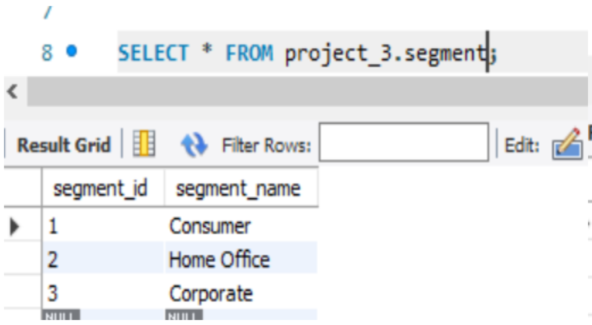
```sql
SELECT * FROM project_3.category;
```

category_id	category_name
1	Office Supplies
2	Furniture
3	Technology
NULL	NULL



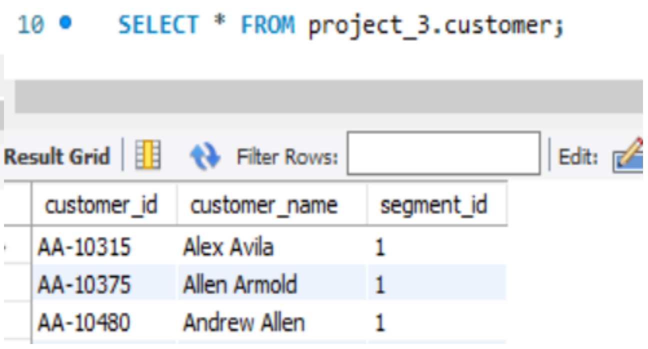
```sql
SELECT * FROM project_3.product;
```

product_id	product_name	category_id
FUR-ADV-10002601	Advantus Photo Frame Erganomic	2
FUR-BO-10000259	Safco Classic Bookcase Traditional	2
FUR-FU-10003447	Eldon Light Bulb Duo Pack	2
OFF-AP-10001254	KitchenAid Coffee Grinder Red	1
OFF-AP-10002317	Hamilton Beach Refrigerator Silver	1



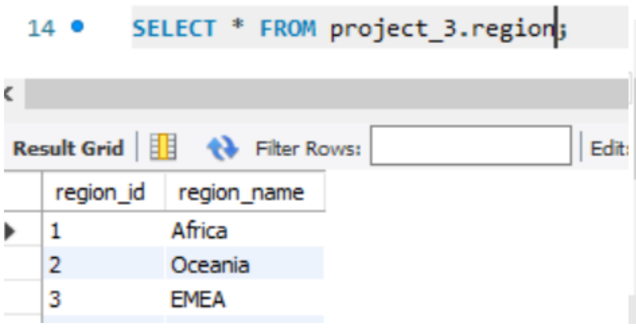
```sql
SELECT * FROM project_3.segment;
```

segment_id	segment_name
1	Consumer
2	Home Office
3	Corporate



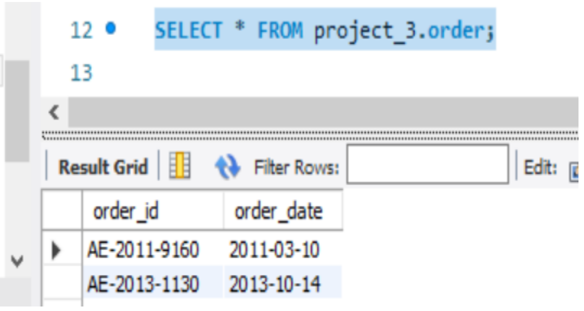
```sql
SELECT * FROM project_3.customer;
```

customer_id	customer_name	segment_id
AA-10315	Alex Avila	1
AA-10375	Allen Armold	1
AA-10480	Andrew Allen	1



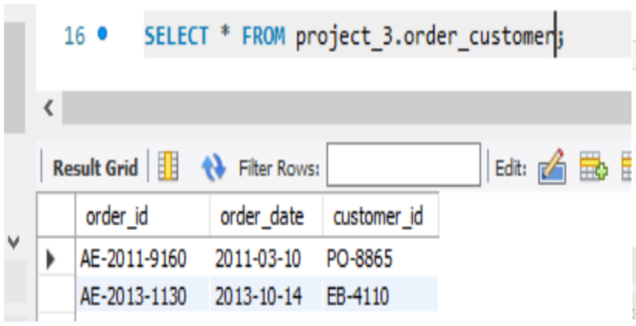
```sql
SELECT * FROM project_3.region;
```

region_id	region_name
1	Africa
2	Oceania
3	EMEA



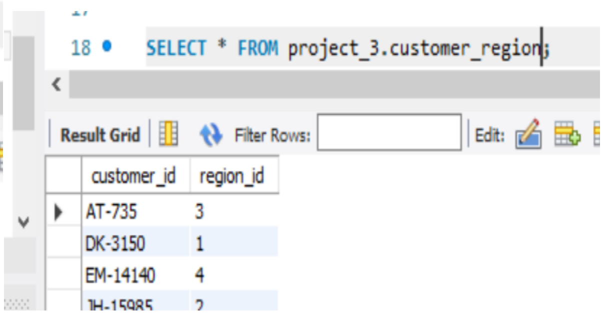
```sql
SELECT * FROM project_3.order;
```

order_id	order_date
AE-2011-9160	2011-03-10
AE-2013-1130	2013-10-14



```sql
SELECT * FROM project_3.order_customer;
```

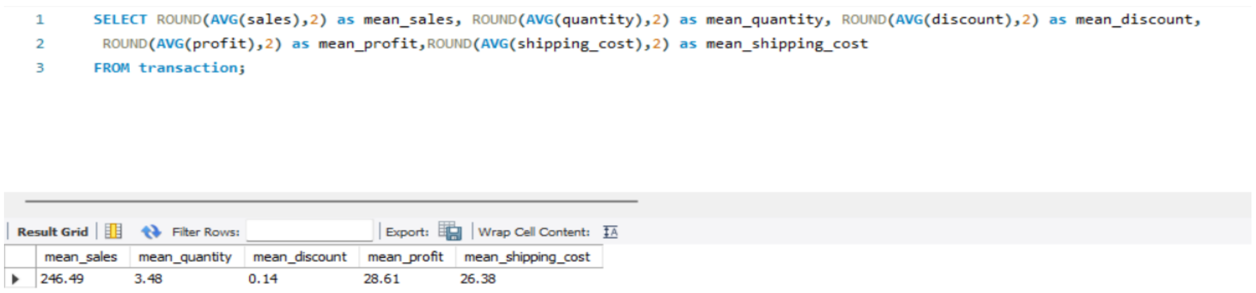
order_id	order_date	customer_id
AE-2011-9160	2011-03-10	PO-8865
AE-2013-1130	2013-10-14	EB-4110



```sql
SELECT * FROM project_3.customer_region;
```

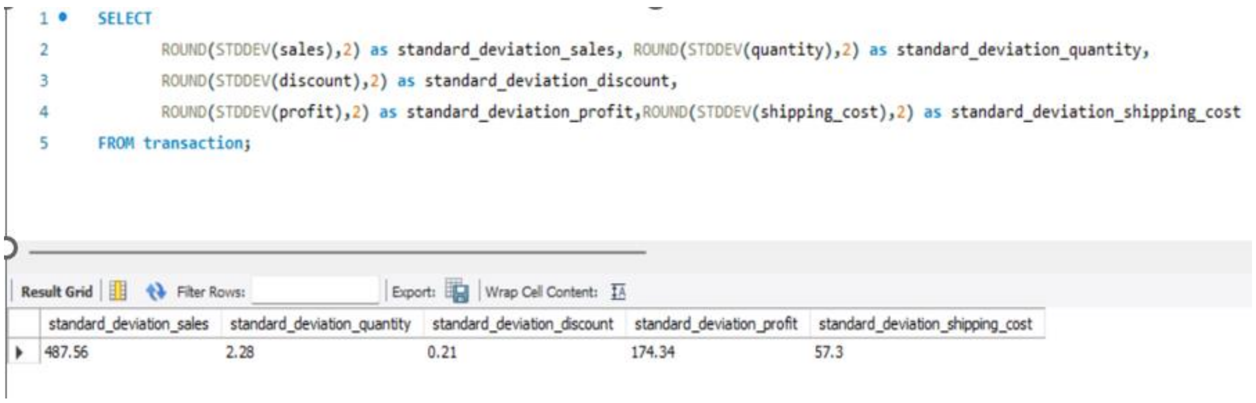
customer_id	region_id
AT-735	3
DK-3150	1
EM-14140	4
JH-15985	2

Statistics:



```sql
SELECT ROUND(AVG(sales),2) as mean_sales, ROUND(AVG(quantity),2) as mean_quantity, ROUND(AVG(discount),2) as mean_discount,
 ROUND(AVG(profit),2) as mean_profit,ROUND(AVG(shipping_cost),2) as mean_shipping_cost
FROM transaction;
```

```
mean_sales  mean_quantity  mean_discount  mean_profit  mean_shipping_cost
246.49      3.48           0.14           28.61        26.38
```



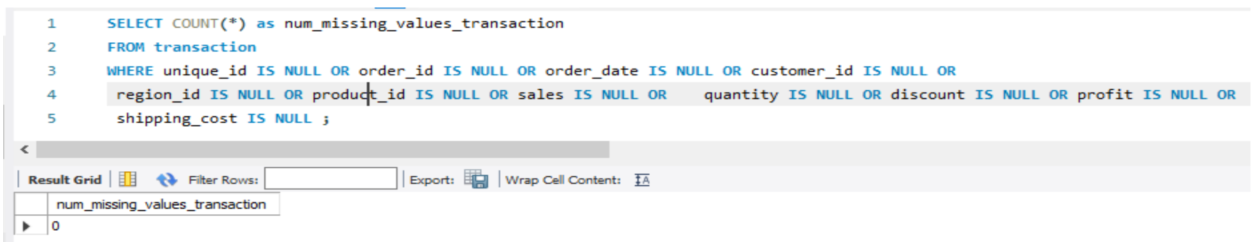
```sql
SELECT
        ROUND(STDDEV(sales),2) as standard_deviation_sales, ROUND(STDDEV(quantity),2) as standard_deviation_quantity,
        ROUND(STDDEV(discount),2) as standard_deviation_discount,
        ROUND(STDDEV(profit),2) as standard_deviation_profit,ROUND(STDDEV(shipping_cost),2) as standard_deviation_shipping_cost
FROM transaction;
```

```
standard_deviation_sales  standard_deviation_quantity  standard_deviation_discount  standard_deviation_profit  standard_deviation_shipping_cost
487.56                    2.28                         0.21                         174.34                     57.3
```

For the numeric columns, mean and standard deviation were found and they were same as the previous results found in step 3 of project 1.

Missing Values:

From the below, it can be inferred that there are no missing values in the transaction table, if there would have been output more than 0, it would have required further evaluation to find which column has null values. Similar queries were run to find missing values and there are no missing values in other tables (~ columns), SQL queries are provided in the appendix.

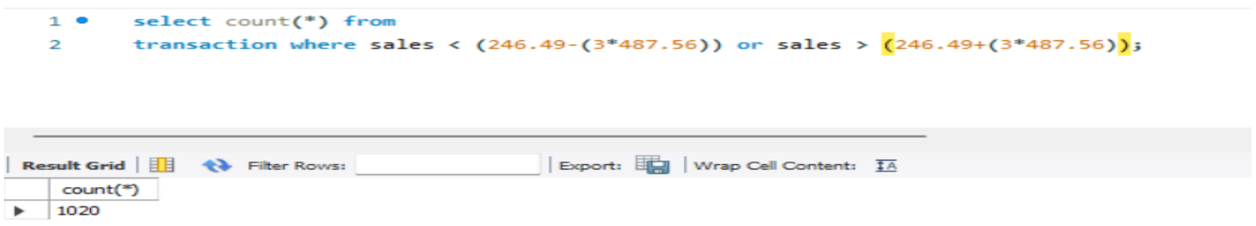


```sql
SELECT COUNT(*) as num_missing_values_transaction
FROM transaction
WHERE unique_id IS NULL OR order_id IS NULL OR order_date IS NULL OR customer_id IS NULL OR
 region_id IS NULL OR product_id IS NULL OR sales IS NULL OR    quantity IS NULL OR discount IS NULL OR profit IS NULL OR
 shipping_cost IS NULL ;
```

```
num_missing_values_transaction
0
```

Outliers:

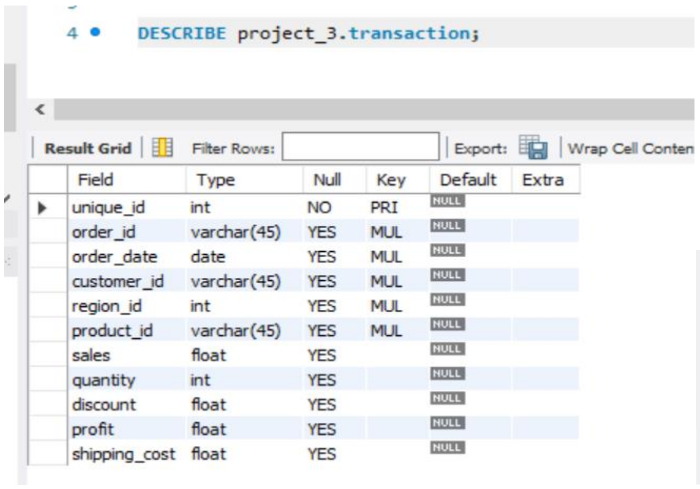
For all numeric columns, we have outliers, For sales column 1020, quantity column 601, discount column 318, profit column 865 and shipping cost 1039 outliers are there. Outliers are found for the sales column below and details of others are provided in the appendix.



```
select count(*) from
transaction where sales < (246.49-(3*487.56)) or sales > (246.49+(3*487.56));
```

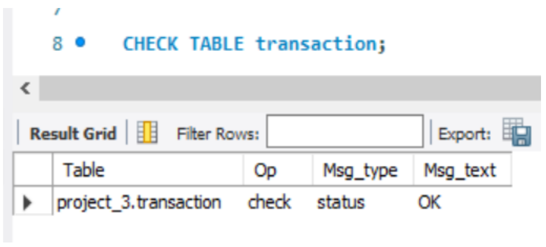
```
count(*)
1020
```

The transaction table contains numeric columns, and we can see from the results there are no data errors and no invalid values.



```
DESCRIBE project_3.transaction;
```

Field	Type	Null	Key	Default	Extra
unique_id	int	NO	PRI	NULL	
order_id	varchar(45)	YES	MUL	NULL	
order_date	date	YES	MUL	NULL	
customer_id	varchar(45)	YES	MUL	NULL	
region_id	int	YES	MUL	NULL	
product_id	varchar(45)	YES	MUL	NULL	
sales	float	YES		NULL	
quantity	int	YES		NULL	
discount	float	YES		NULL	
profit	float	YES		NULL	
shipping_cost	float	YES		NULL	

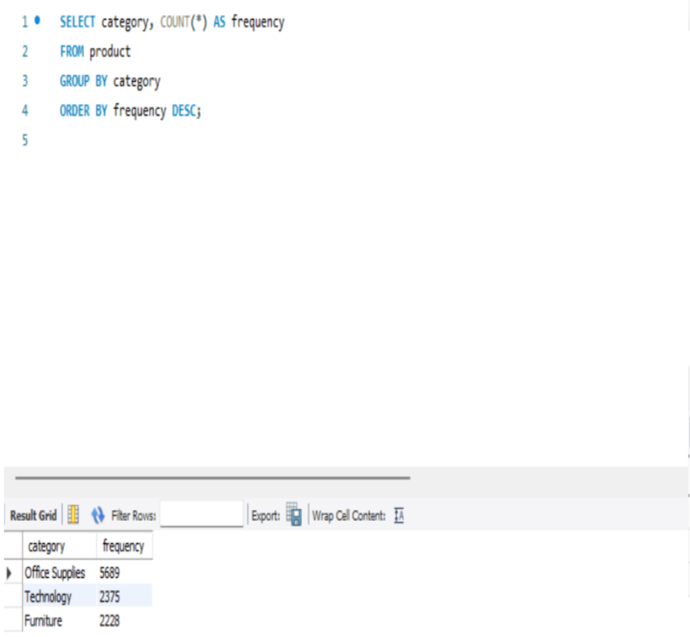


```
CHECK TABLE transaction;
```

Table	Op	Msg_type	Msg_text
project_3.transaction	check	status	OK

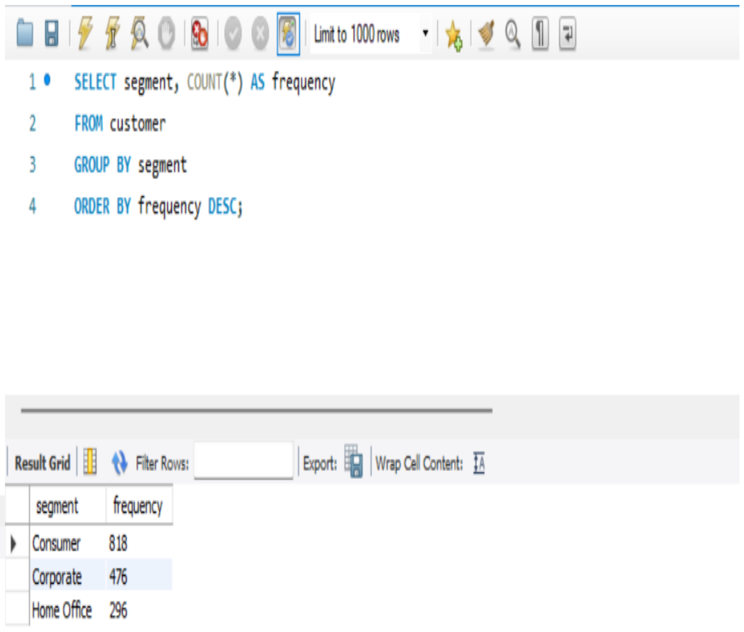
Frequency table, characters column:

For category, segment, region columns, frequency was found, and verified for any data errors, from the below screenshot it can be concluded that there are no data errors.



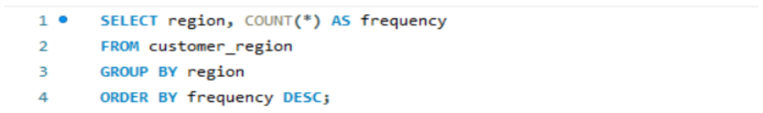
```
SELECT category, COUNT(*) AS frequency
FROM product
GROUP BY category
ORDER BY frequency DESC;
```

category	frequency
Office Supplies	5689
Technology	2375
Furniture	2228

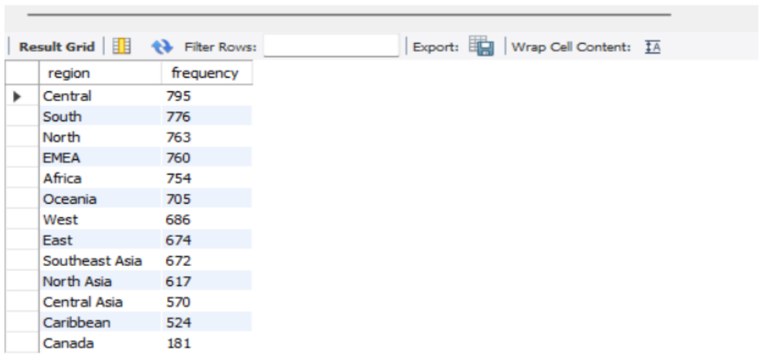


```
SELECT segment, COUNT(*) AS frequency
FROM customer
GROUP BY segment
ORDER BY frequency DESC;
```

segment	frequency
Consumer	818
Corporate	476
Home Office	296



```
SELECT region, COUNT(*) AS frequency
FROM customer_region
GROUP BY region
ORDER BY frequency DESC;
```



region	frequency
Central	795
South	776
North	763
EMEA	760
Africa	754
Oceania	705
West	686
East	674
Southeast Asia	672
North Asia	617
Central Asia	570
Caribbean	524
Canada	181

From the above query outputs and by checking clearly there are no data errors and also there are no values that seem to be invalid.

A similar query which is used to find missing value for a transaction table, mentioned above is used to find the missing values for character columns, details of the same are provided in appendix.

All values that are available in the table for character columns are clearly of same data type and domain.

Join queries:

The following join queries are executed and screenshots are attached below.

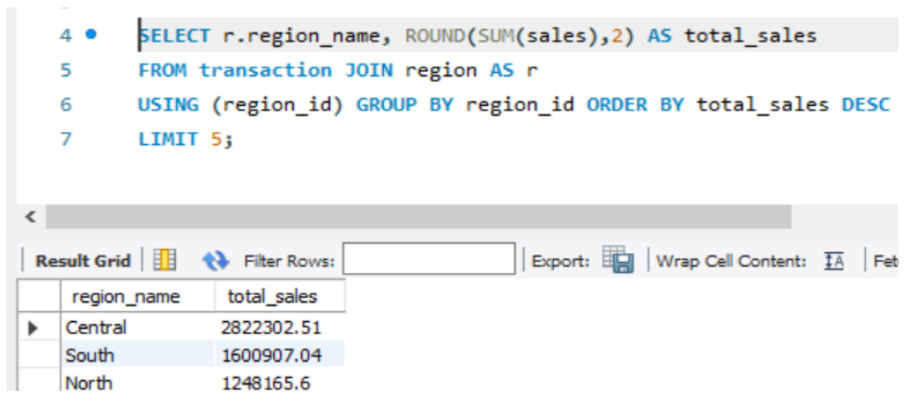
1. Region and transaction tables are joined and found region wise total sales, region and transaction are joined on region_id, and grouped by region_id to find the total sales.

2. product, category and transaction tables are joined and found category wise total profit, initially product and transaction are joined on product_id, this is further joined with category table using category _id and grouped by category_id to find the total profit.

3. customer and transaction are joined to find the name of the customer who has max sales, customer and transaction table are joined on customer_id, grouped by customer_id to find the total sales.

All codes are successfully executed and output of the queries are valid.

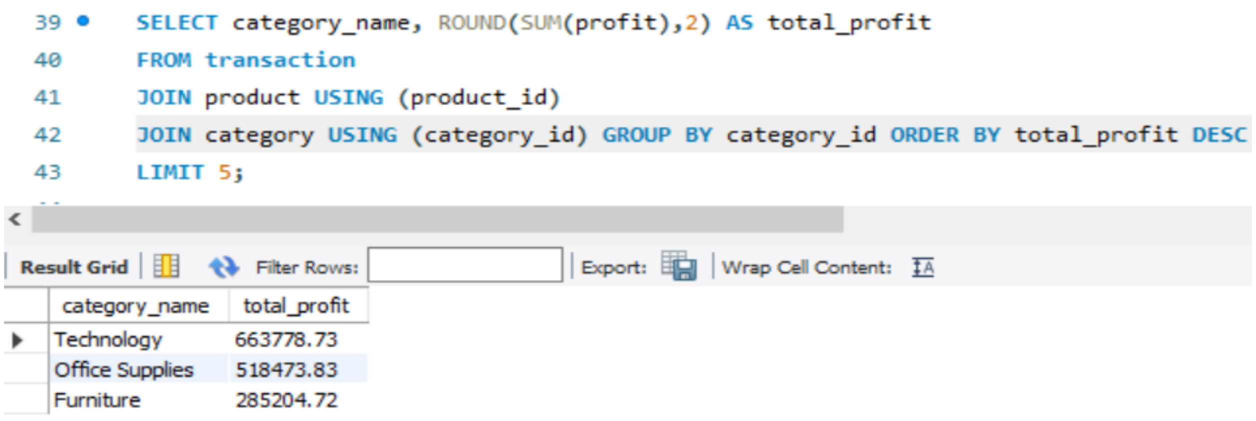
First Join query:



```
SELECT r.region_name, ROUND(SUM(sales),2) AS total_sales
FROM transaction JOIN region AS r
USING (region_id) GROUP BY region_id ORDER BY total_sales DESC
LIMIT 5;
```

```
Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fet
region_name   total_sales
Central       2822302.51
South         1600907.04
North         1248165.6
```

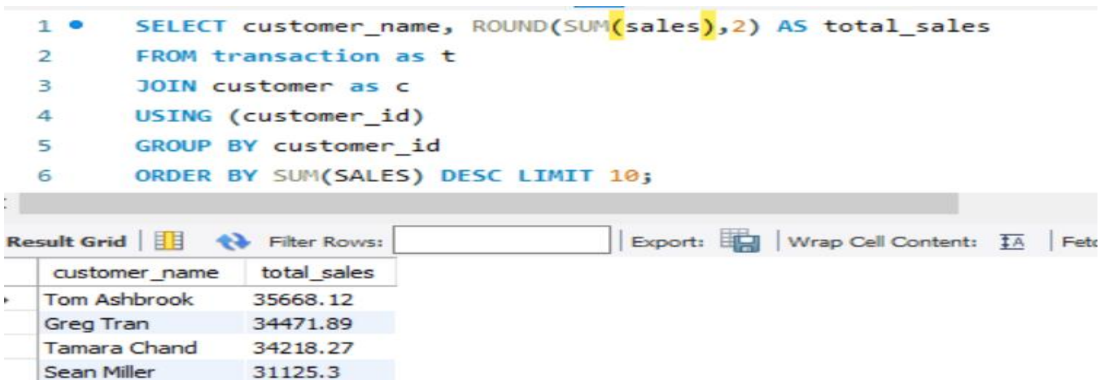
Second Join query:



```
SELECT category_name, ROUND(SUM(profit),2) AS total_profit
FROM transaction
JOIN product USING (product_id)
JOIN category USING (category_id) GROUP BY category_id ORDER BY total_profit DESC
LIMIT 5;
```

```
Result Grid | Filter Rows: | Export: | Wrap Cell Content:
category_name     total_profit
Technology        663778.73
Office Supplies   518473.83
Furniture         285204.72
```

Third Join query:



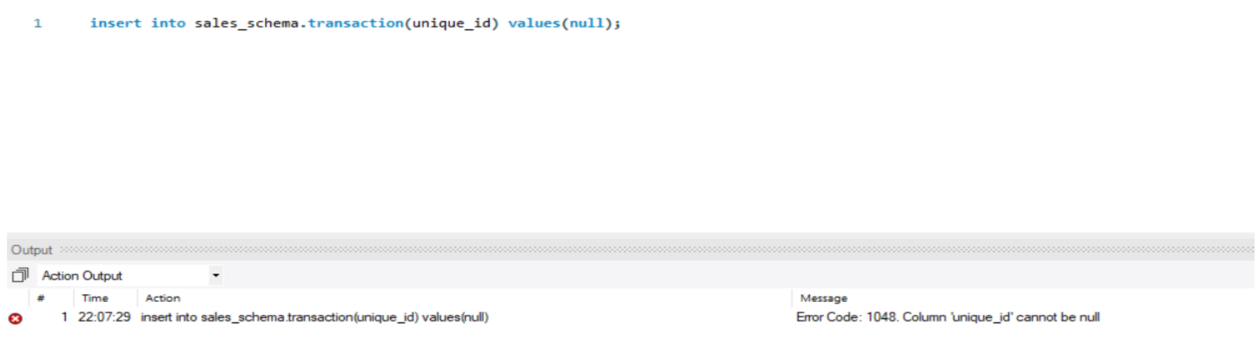
```
SELECT customer_name, ROUND(SUM(sales),2) AS total_sales
FROM transaction as t
JOIN customer as c
USING (customer_id)
GROUP BY customer_id
ORDER BY SUM(SALES) DESC LIMIT 10;
```

```
Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetc
customer_name    total_sales
Tom Ashbrook     35668.12
Greg Tran        34471.89
Tamara Chand     34218.27
Sean Miller      31125.3
```

Constraints:

Primary Key Constraint:

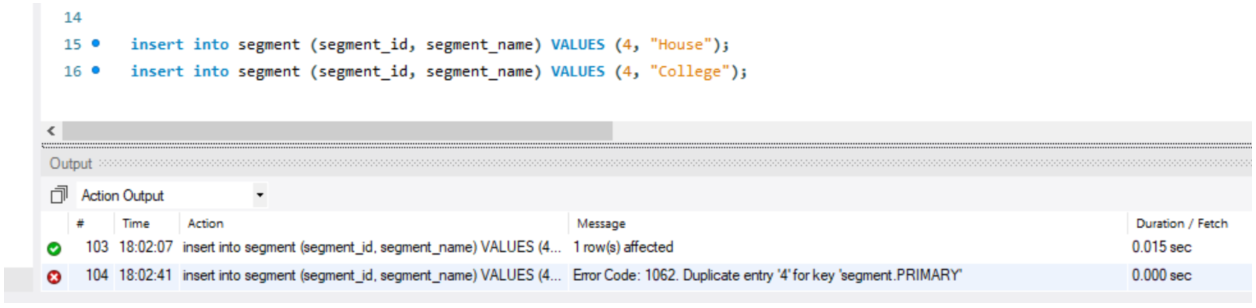
From the below query output it is observed that due to primary key constraint transaction table is not accepting null value for unique_id which is a primary key for transaction table.



```sql
insert into sales_schema.transaction(unique_id) values(null);
```

```
Output
Action Output
#  Time      Action                                                   Message
1  22:07:29  insert into sales_schema.transaction(unique_id) values(null)   Error Code: 1048. Column 'unique_id' cannot be null
```

Unique Key Constraint:

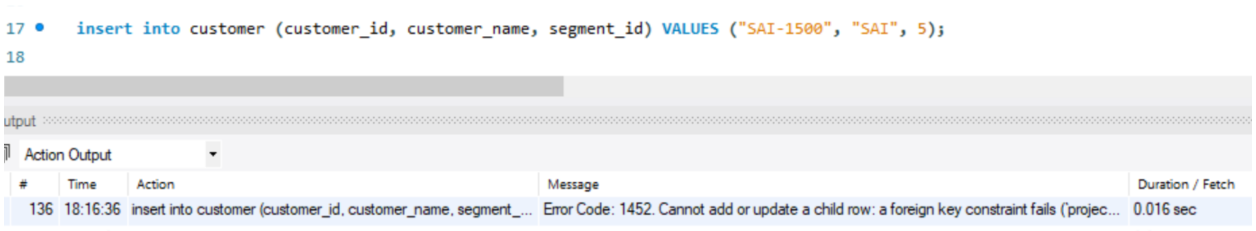


```sql
insert into segment (segment_id, segment_name) VALUES (4, "House");
insert into segment (segment_id, segment_name) VALUES (4, "College");
```

```
Output
Action Output
#    Time      Action                                                   Message                                                       Duration / Fetch
103  18:02:07  insert into segment (segment_id, segment_name) VALUES (4...  1 row(s) affected                                             0.015 sec
104  18:02:41  insert into segment (segment_id, segment_name) VALUES (4...  Error Code: 1062. Duplicate entry '4' for key 'segment.PRIMARY'  0.000 sec
```

From the query output we can observe that due to primary key, segment_id doesn't accept duplicate values.

Referential Integrity Constraint:

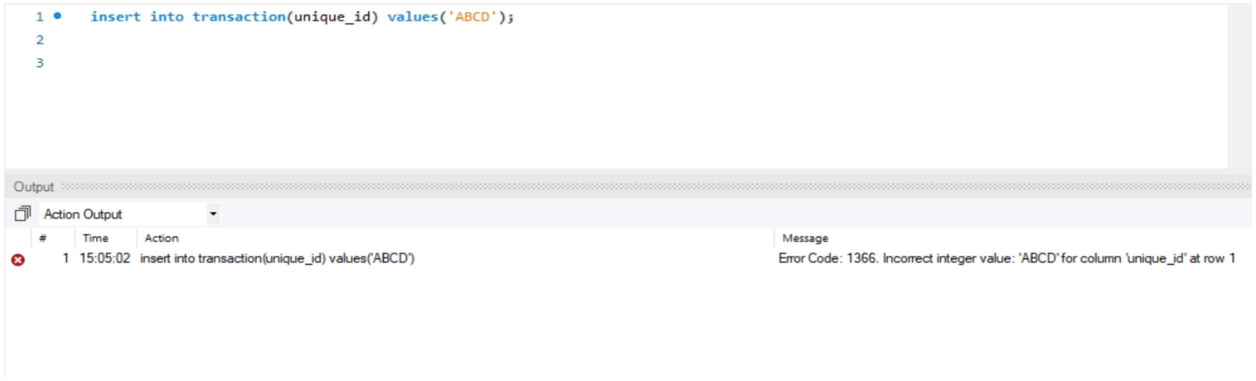


```sql
insert into customer (customer_id, customer_name, segment_id) VALUES ("SAI-1500", "SAI", 5);
```

```
utput
Action Output
#    Time      Action                                                   Message                                                                            Duration / Fetch
136  18:16:36  insert into customer (customer_id, customer_name, segment_...  Error Code: 1452. Cannot add or update a child row: a foreign key constraint fails ('projec...  0.016 sec
```

From the output it is observed that foreign key constraint is working correctly and preventing the insertion of a new segment_id, because segment_id is a foreign key in customer table.

Domain Constraint:



```sql
insert into transaction(unique_id) values('ABCD');
```

```
Output
Action Output
#  Time      Action                                            Message
1  15:05:02  insert into transaction(unique_id) values('ABCD')   Error Code: 1366. Incorrect integer value: 'ABCD' for column 'unique_id' at row 1
```

Since 'unique_id' data type is INT, the value being inserted ('ABCD') is not a valid integer value.