Course D597 Task 1 Scenario 2

A.

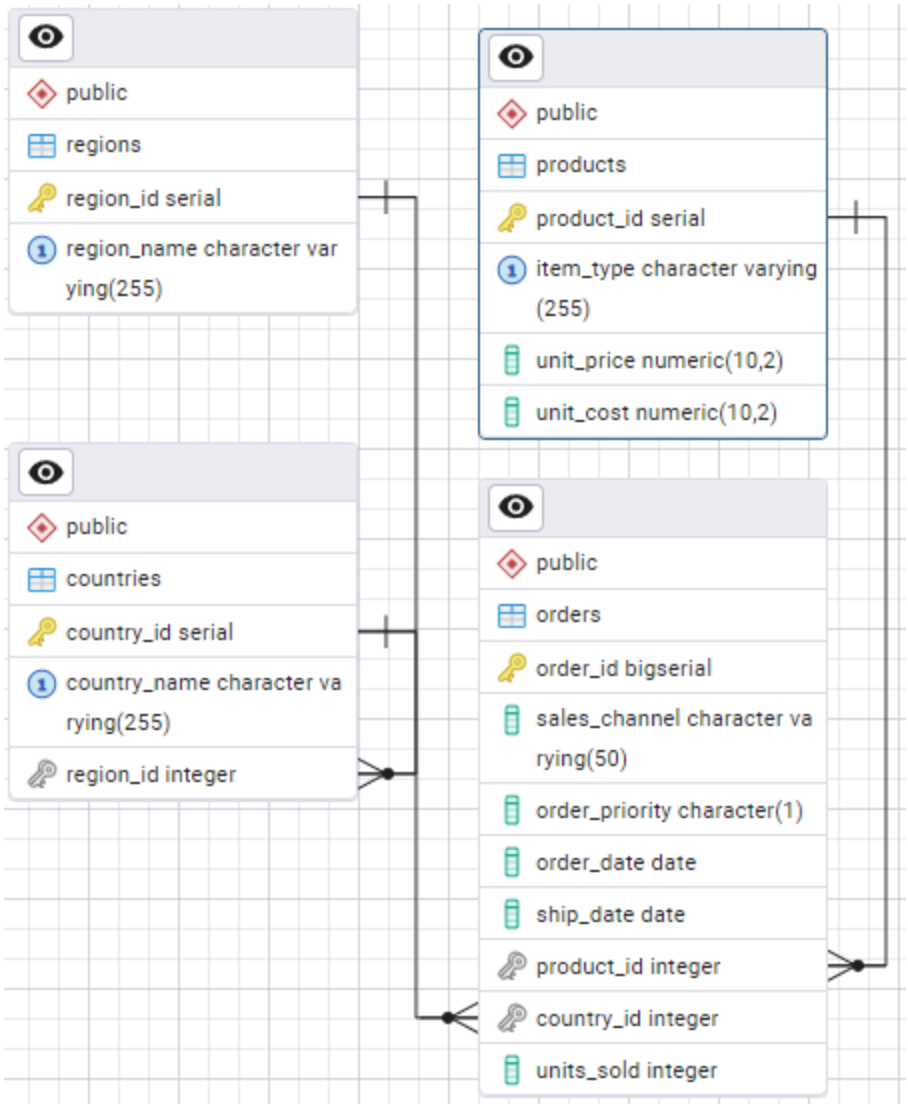
1. A business problem that EcoMart has that can be solved with a database solution is that Ecomart handles a lot of customer transactions and needs a flexible and scalable way to store and analyze that data without jeopardizing data integrity or query performance. We could imagine the marketing department requires data to optimize a new marketing campaign to increase revenue. We could use provide various pieces of previous order data to assist the marketing team in their decisions around this campaign.

2. This data should be put into a Snowflake Schema.

3. Putting the data into a snowflake schema will organize all the data in a way where queries remain simple, data integrity is well protected, and the database is scalable, as only the orders table needs to be added to as more orders come in. Only the products table needs to be altered if products are added to the product base.

4. While assisting the marketing department in organizing a campaign to increase revenue, they will need to know data points such as what products are selling best in each different region, what regions are generating the most/least revenue on the Ecomart platform, and what products have been the most profitable. This data is easily accessible in this data structure and will provide the marketing team with some of the necessary insight to design their marketing campaign.

B. .



C. My database is in 3NF. The database consists of 4 tables: Regions, Countries, Products, and Orders.
Regions will hold all region names in the data and assign each region a unique ID.
Countries will hold all of the country names and assign each country a unique ID. The country table will also relate each country to its respective region via the foreign key link to the Regions table.
Products will hold all of the product types and assign each one a unique ID. This table will also hold the unit price and unit cost of each individual product type.
Orders will hold the order ID present in the original dataset, as well as the sales channel, the order priority, the order data, the ship date, and the number of units sold. It will also relate each order to its respective country and product through the foreign key links to the Countries and Product tables.

D. The proposed database design handles the scalability mentioned in the scenario by limiting the number of tables that need to be added to as the sales volume and diverse product base grow. In a flat table, every time a new order was made, a new row would be created, leading to a large amount of data, including region, country, and product data, being replicated over and over in the table, increasing the strain on the data storage solutions. By normalizing the data in the way I have, only the orders table will need to be updated when a new order is made and only the product table will need to be updated when new products are added to the product base.

E. When implementing this database design, it would be important to carry out privacy and security measures to protect sensitive data from being released, and to protect the integrity of the data. I do not believe any of the current database necessarily warrants serious encryption measures; however, if we began storing customer or employee data, it would be vital to introduce encryption strategies to make personal data less accessible. I would ensure that access to the database is placed behind 2-Factor Authentication to verify the identity of any user.
I would also ensure that only the necessary teams have access to read privileges on the required data, and an even smaller group has write privileges, to ensure only those with adequate experience can add and remove records from the database. We can maintain data integrity by limiting the ability to edit the database to a small number of experienced team members. I would also install a backup system that updates regularly to ensure that the database can easily be returned to its former state if any improper changes are made.

F.

1. Below is a screenshot of the script used to create the database proposed in Part B.

```
CREATE DATABASE "D597 Task 1"
    WITH
    OWNER = postgres
    ENCODING = 'UTF8'
    LOCALE_PROVIDER = 'libc'
    CONNECTION LIMIT = -1
    IS_TEMPLATE = False;
```

```sql
CREATE TABLE regions (
    region_id SERIAL PRIMARY KEY,
    region_name VARCHAR(255) UNIQUE NOT NULL
);

CREATE TABLE countries (
    country_id SERIAL PRIMARY KEY,
    country_name VARCHAR(255) UNIQUE NOT NULL,
    region_id INT NOT NULL,
    FOREIGN KEY (region_id) REFERENCES regions(region_id)
);

CREATE TABLE products (
    product_id SERIAL PRIMARY KEY,
    item_type VARCHAR(255) UNIQUE NOT NULL,
    unit_price NUMERIC(10,2) NOT NULL,
    unit_cost NUMERIC(10,2) NOT NULL
);

CREATE TABLE orders (
    order_id BIGSERIAL PRIMARY KEY,
    sales_channel VARCHAR(50) NOT NULL CHECK (sales_channel IN ('Online', 'Offline')),
    order_priority CHAR(1) NOT NULL CHECK (order_priority IN ('L', 'M', 'H', 'C')),
    order_date DATE NOT NULL,
    ship_date DATE NOT NULL,
    product_id INT NOT NULL,
    country_id INT NOT NULL,
    units_sold INT NOT NULL CHECK (units_sold >= 0),
    FOREIGN KEY (product_id) REFERENCES products(product_id),
    FOREIGN KEY (country_id) REFERENCES countries(country_id)
);
```

2. Below is a script to import all of the data from the provided CSV into my database.

```
CREATE TEMP TABLE staging_sales (
    region_name VARCHAR(255),
    country_name VARCHAR(255),
    item_type VARCHAR(255),
    sales_channel VARCHAR(50),
    order_priority CHAR(1),
    order_date DATE,
    order_id BIGINT,
    ship_date DATE,
    units_sold INT,
    unit_price NUMERIC(10,2),
    unit_cost NUMERIC(10,2)
);

COPY staging_sales FROM E'C:\\temp\\UpdatedSalesData.csv'
DELIMITER ',' ;

INSERT INTO regions (region_name)
SELECT DISTINCT region_name FROM staging_sales
ORDER BY region_name;

INSERT INTO countries (country_name, region_id)
SELECT DISTINCT s.country_name, r.region_id
FROM staging_sales s
JOIN regions r ON s.region_name = r.region_name
ORDER BY country_name;

INSERT INTO products (item_type, unit_price, unit_cost)
SELECT DISTINCT item_type, unit_price, unit_cost
FROM staging_sales
ORDER BY item_type;

INSERT INTO orders (order_id, sales_channel, order_priority, order_date, ship_date, units_sold, product_id, country_id)
SELECT s.order_id, s.sales_channel, s.order_priority, s.order_date, s.ship_date, s.units_sold, p.product_id, c.country_id
FROM staging_sales s
JOIN products p ON s.item_type = p.item_type
JOIN countries c ON s.country_name = c.country_name
ORDER BY order_id;

DROP TABLE staging_sales;
```

Below are screenshots of the outputs of all the tables present in the database.

country_id [PK] integer	country_name character varying (255)	region_id integer
1	Afghanistan	5
2	Albania	4
3	Algeria	5
4	Andorra	4
5	Angola	7
6	Antigua and Barbuda	3
7	Armenia	4
8	Australia	2
9	Austria	4
10	Azerbaijan	5
11	Bahrain	5
12	Bangladesh	1
13	Barbados	3
14	Belarus	4
15	Belgium	4
16	Belize	3
17	Benin	7
18	Bhutan	1

product_id [PK] integer	item_type character varying (255)	unit_price numeric (10,2)	unit_cost numeric (10,2)
14	Baby Food	255.28	159.42
15	Beverages	47.45	31.79
16	Cereal	205.70	117.11
17	Clothes	109.28	35.84
18	Cosmetics	437.20	263.33
19	Fruits	9.33	6.92
20	Household	668.27	502.54
21	Meat	421.89	364.69
22	Office Supplies	651.21	524.96
23	Personal Care	81.73	56.67
24	Snacks	152.58	97.44
25	Vegetables	154.06	90.93

region_id [PK] integer	region_name character varying (255)
1	Asia
2	Australia and Oceania
3	Central America and the Caribbean
4	Europe
5	Middle East and North Africa
6	North America
7	Sub-Saharan Africa

order_id [PK] bigint	sales_channel character varying (50)	order_priority character (1)	order_date date	ship_date date	product_id integer	country_id integer	units_sold integer
100008904	Offline	L	2015-10-19	2015-12-08	17	137	3712
100009763	Offline	M	2012-07-29	2012-09-08	21	44	3966
100035941	Offline	C	2016-07-16	2016-08-15	21	185	1713
100043666	Offline	L	2010-04-06	2010-04-09	19	94	3999
100050961	Online	H	2016-12-14	2017-01-14	23	7	6158
100051820	Online	C	2013-09-24	2013-10-16	25	13	6412
100054824	Online	C	2013-10-24	2013-12-06	14	77	7301
100062119	Online	L	2012-12-06	2012-12-26	22	174	9460
100069415	Online	M	2012-01-19	2012-03-07	17	63	1619
100077140	Online	C	2013-05-06	2013-05-26	23	2	3904
100088727	Online	M	2011-07-02	2011-07-06	20	27	7333
100089585	Online	C	2015-11-07	2015-12-24	22	99	7587
100105893	Online	C	2015-03-21	2015-04-24	17	114	2413
100106751	Online	L	2011-12-30	2012-01-24	21	68	2667
100109755	Online	L	2012-01-29	2012-03-15	16	134	3556
100110614	Online	M	2016-06-06	2016-07-13	24	164	3810
100114047	Online	H	2011-02-11	2011-02-13	24	150	4826
100147092	Offline	C	2012-01-09	2012-02-02	24	121	4605
100155675	Offline	H	2010-02-04	2010-03-15	25	141	7145
100165116	Offline	H	2012-07-08	2012-08-22	23	14	9939
100166833	Offline	L	2013-08-24	2013-09-20	16	140	447
100176274	Offline	L	2016-01-26	2016-02-28	22	184	3241
100177562	Offline	H	2015-01-09	2015-01-29	21	112	3622

3. Below are three queries and outputs that could be used to solve the business problem presented in Part A1 using the business data solutions suggested in Part A4.

```sql
WITH product_sales AS (
SELECT r.region_name,
p.item_type,
SUM(o.units_sold) AS total_units_sold,
RANK() OVER (PARTITION BY r.region_name ORDER BY SUM(o.units_sold) DESC) AS RANK
FROM orders o
JOIN products p ON o.product_id = p.product_id
JOIN countries c ON o.country_id = c.country_id
JOIN regions r ON c.region_id = r.region_id
GROUP BY r.region_name, p.item_type
)
SELECT region_name, item_type, total_units_sold
FROM product_sales
WHERE rank = 1;
```

	region_name character varying (255)	item_type character varying (255)	total_units_sold bigint
1	Asia	Cereal	6397658
2	Australia and Oceania	Personal Care	3616014
3	Central America and the Caribbean	Cosmetics	4814795
4	Europe	Cereal	11076271
5	Middle East and North Africa	Office Supplies	5431214
6	North America	Cosmetics	996810
7	Sub-Saharan Africa	Cosmetics	11318369

Total rows: 7 Query complete 00:00:00.343

```sql
SELECT r.region_name,
SUM(o.units_sold * p.unit_price) AS total_revenue
FROM orders o
JOIN products p ON o.product_id = p.product_id
JOIN countries c ON o.country_id = c.country_id
JOIN regions r ON c.region_id = r.region_id
GROUP BY region_name
ORDER BY total_revenue DESC;
```

	region_name character varying (255)	total_revenue numeric
1	Sub-Saharan Africa	34958453406.17
2	Europe	34241150923.39
3	Asia	19293401219.82
4	Middle East and North Africa	16921412794.52
5	Central America and the Caribbean	14553730165.29
6	Australia and Oceania	10701522223.73
7	North America	2937002333.49

Total rows: 7 | Query complete 00:00:00.693

```sql
SELECT p.item_type,
SUM(o.units_sold * (p.unit_price - p.unit_cost)) AS total_profit
FROM orders o
JOIN products p ON o.product_id = p.product_id
GROUP BY p.item_type
ORDER BY total_profit DESC;
```

	item_type character varying (255)	total_profit numeric
1	Cosmetics	7289406555.68
2	Household	6870966095.35
3	Office Supplies	5339532912.50
4	Baby Food	4017647893.20
5	Cereal	3743318890.62
6	Clothes	3067841433.60
7	Vegetables	2604417796.68
8	Meat	2387834992.40
9	Snacks	2299287932.88
10	Personal Care	1040435215.96
11	Beverages	650112575.58
12	Fruits	98321435.16

Total rows: 12 | Query complete 00:00:00.138

4. These queries can be optimized by creating indexes on frequently used columns. In this case, I have indexed the foreign keys present in the orders table, the units_sold column of the orders table because it is present in all three queries, and the unit_price column of the products table because it is used in calculations in 2 of the queries.

```
CREATE INDEX idx_orders_product_id ON orders(product_id);
CREATE INDEX idx_orders_country_id ON orders(country_id);
CREATE INDEX idx_countries_region_id ON countries(region_id);
CREATE INDEX idx_orders_units_sold ON orders(units_sold);
CREATE INDEX idx_products_unit_price ON products(unit_price);
```

Below are screenshots of the outputs of all three queries and their completion times after the indexes have been created.

	region_name character varying (255)	item_type character varying (255)	total_units_sold bigint
1	Asia	Cereal	6397658
2	Australia and Oceania	Personal Care	3616014
3	Central America and the Caribbean	Cosmetics	4814795
4	Europe	Cereal	11076271
5	Middle East and North Africa	Office Supplies	5431214
6	North America	Cosmetics	996810
7	Sub-Saharan Africa	Cosmetics	11318369

Total rows: 7 Query complete 00:00:00.075

	region_name character varying (255)	total_revenue numeric
1	Sub-Saharan Africa	34958453406.17
2	Europe	34241150923.39
3	Asia	19293401219.82
4	Middle East and North Africa	16921412794.52
5	Central America and the Caribbean	14553730165.29
6	Australia and Oceania	10701522223.73
7	North America	2937002333.49

Total rows: 7 Query complete 00:00:00.094

	item_type character varying (255)	total_profit numeric
1	Cosmetics	7289406555.68
2	Household	6870966095.35
3	Office Supplies	5339532912.50
4	Baby Food	4017647893.20
5	Cereal	3743318890.62
6	Clothes	3067841433.60
7	Vegetables	2604417796.68
8	Meat	2387834992.40
9	Snacks	2299287932.88
10	Personal Care	1040435215.96
11	Beverages	650112575.58
12	Fruits	98321435.16

Total rows: 12 | Query complete 00:00:00.118

All three queries had their completion times reduced following the optimizations.