Johnson Millil

Data Management

January 13, 2025

Relational Database Design for Ecomart Report

Part 1: Design Document

A1. Business Problem

EcoMart, an eco-conscious online marketplace, faces challenges in managing and analyzing its data effectively (Elmasri & Navathe, 2016; United Nations, 2015). As the platform grows, the company struggles to:

- Track product sales and revenue efficiently across different regions.
- Identify high-performing eco-friendly products based on sustainability certifications.
- Analyze customer reviews to assess product performance and improve satisfaction.
- Maintain scalability to handle increasing data volumes and user traffic.

A well-designed relational database can centralize data, enable advanced querying, and support informed decision-making to address these challenges.

A2. Data Structure

The proposed relational database structure includes the following key tables (PostgreSQL Global Development Group, n.d.):

1. **Products Table**:
    - Captures product details, including sustainability certifications and eco-impact ratings.
2. **Orders Table**:
    - Tracks sales information, such as order IDs, regions, quantities sold, and revenue.
3. **Reviews Table**:
    - Stores user feedback to analyze sentiment and improve product offerings.

Relationships:

- Each product can appear in multiple orders (one-to-many relationship between Products and Orders).
- Each product can have multiple reviews (one-to-many relationship between Products and Reviews).

A3. Database Justification

A relational database offers:

- Data Organization: Centralizes product, sales, and review data in structured tables, making it easier to manage and retrieve.
- Advanced Querying: Enables complex queries to analyze trends, track revenue, and assess customer feedback (Elmasri & Navathe, 2016).
- Scalability: Supports increasing data volumes with indexing, partitioning, and optimization techniques (Stonebraker & Hellerstein, 2005).
- Integration: Allows seamless integration with business analytics tools for real-time insights and reporting.

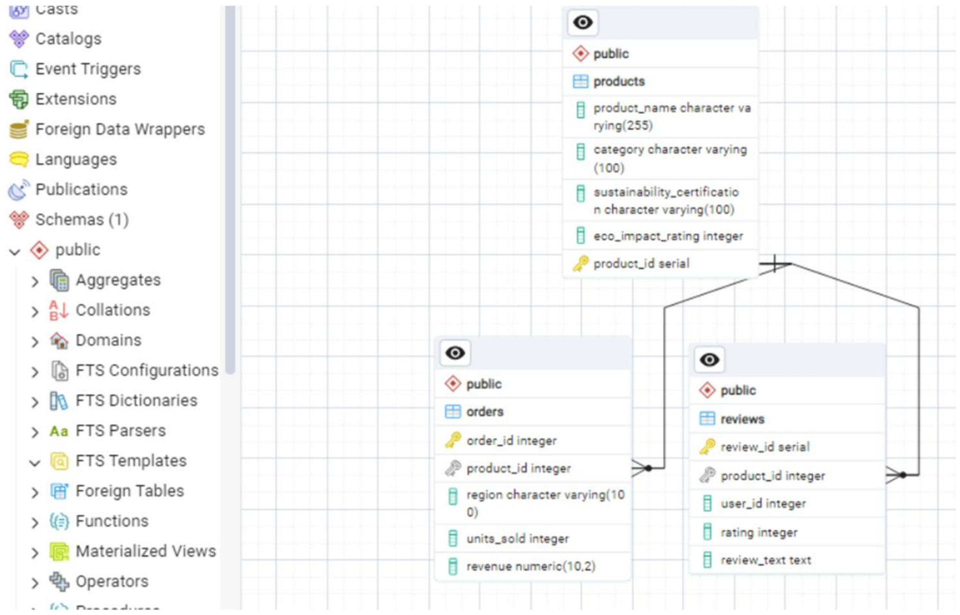
A4. Data Usage

The database will support the following use cases (Pang & Lee, 2008; Elmasri & Navathe, 2016):

1. Product Performance Analysis:
    - Identify top-selling products in specific regions.
    - Compare sales trends across categories (Elmasri & Navathe, 2016; PostgreSQL Global Development Group, n.d.).
2. Customer Feedback Assessment:
    - Analyze reviews to determine customer satisfaction.
    - Identify products requiring improvement (Pang & Lee, 2008).
3. Sustainability Reporting:
    - Generate reports on eco-friendly product performance based on sustainability certifications and ratings.
4. Decision Support:

- Inform inventory management and marketing strategies based on data insights.

B. Logical data model for storing data in the database solution



C. Description of Database Objects and Storage with File Attributes

Database Objects:

1. Tables:
   - Products: Stores product details.
   - Orders: Tracks sales data.
   - Reviews: Holds customer feedback.
2. Indexes:
   - Created on frequently queried fields (Product_ID, Region) to optimize performance.
3. Foreign Keys:
   - Ensure relational integrity between Products, Orders, and Reviews.

File Attributes:

1. Products:
   - Product_Name (String): Maximum length of 255 characters.
   - Category (String): Maximum length of 100 characters.
   - Sustainability_Certification (String): Maximum length of 100 characters.
   - Eco_Impact_Rating (Integer): Range 1–5.
2. Orders:

    - Region (String): Maximum length of 100 characters.
    - Units_Sold (Integer): Positive values only.
    - Revenue (Decimal): Two decimal places for monetary values.
3. Reviews:
    - Rating (Integer): Range 1–5.
    - Review_Text (Text): Unlimited length.

D. Scalability Strategies

Normalization:

- The database is normalized to 3NF (Third Normal Form), reducing redundancy and improving data consistency (Stonebraker & Hellerstein, 2005).

Indexing:

- Frequently queried fields (Product_ID, Region) are indexed to enhance query performance (PostgreSQL Global Development Group, n.d.).

Partitioning:

- Large tables like Orders can be partitioned by region or date for efficient parallel processing.

Horizontal and Vertical Scaling:

- Supports horizontal scaling (distributing data across servers) and vertical scaling (adding resources to the database server).

Caching:

- Frequently accessed data, such as top-selling products, can be cached for quicker retrieval.

E. Privacy and Security Measures

Data Privacy:

1. Encryption:
    - Encrypt sensitive fields (e.g., review text) to protect user data.
    - Use SSL/TLS for secure communication between clients and the database.
2. Anonymization:
    - Replace user-identifiable information (e.g., User_ID) with anonymized IDs.

Access Control:

1. Role-Based Permissions:
   - Grant access to tables and queries based on user roles (e.g., admin, analyst) (PostgreSQL Global Development Group, n.d.).
2. Least Privilege Principle:
   - Limit user access to only the data they need.

Security Measures:

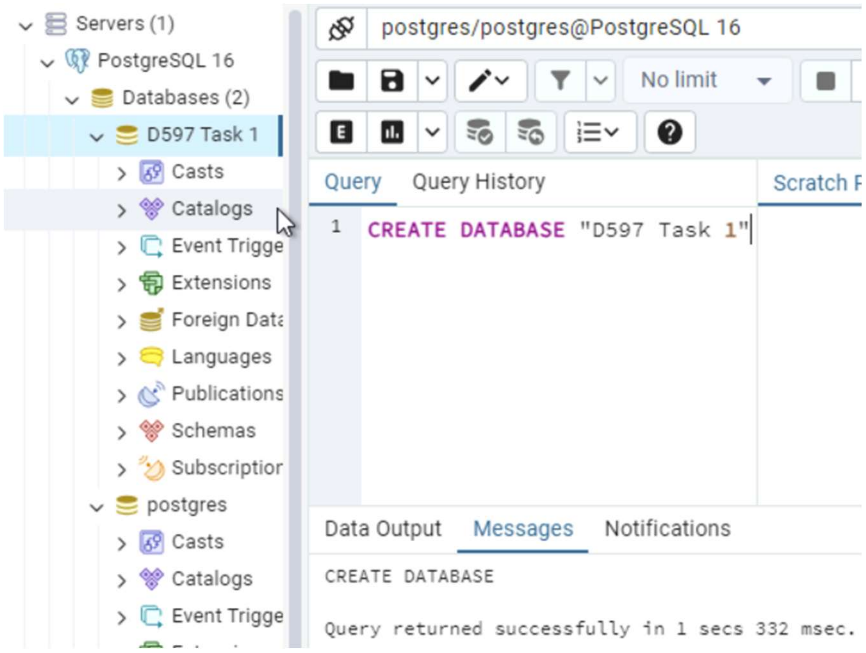
1. Auditing and Logging:
   - Maintain logs of database changes to track unauthorized access or modifications.
2. Authentication:
   - Enforce strong password policies and multi-factor authentication for admin accounts.

Backup and Recovery:

1. Automated Backups:
   - Schedule regular backups of the database.
2. Disaster Recovery:
   - Implement failover servers for high availability in case of system failure.

Part 2: Implementation

F1. Create a Database Instance

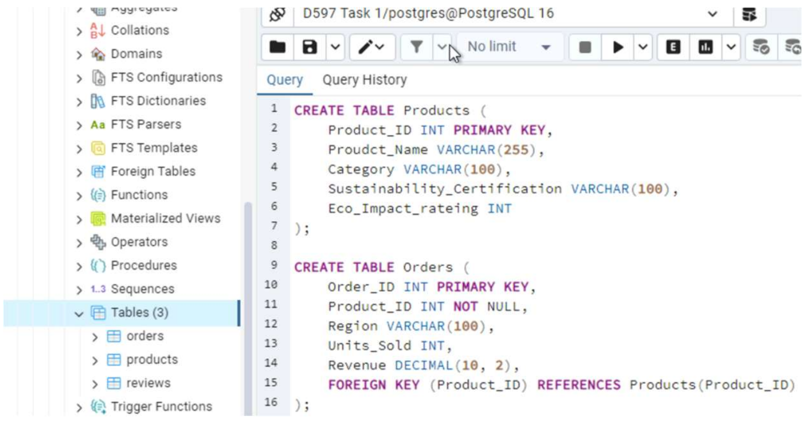


```
CREATE DATABASE "D597 Task 1"
```

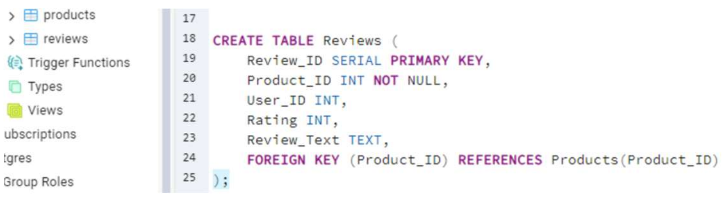
```
CREATE DATABASE

Query returned successfully in 1 secs 332 msec.
```

F2. Create tables & Import Data

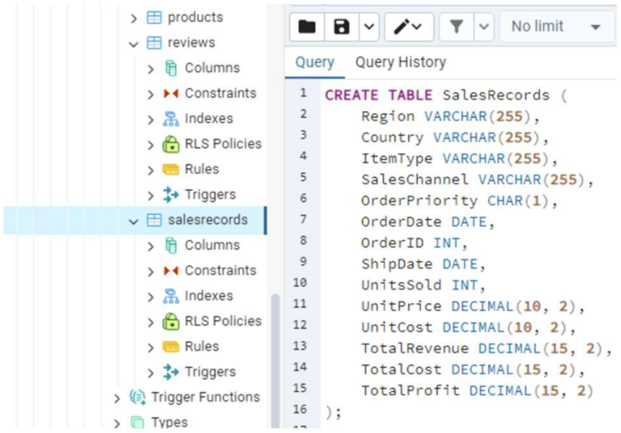
A. Create Tables



```
CREATE TABLE Products (
    Product_ID INT PRIMARY KEY,
    Proudct_Name VARCHAR(255),
    Category VARCHAR(100),
    Sustainability_Certification VARCHAR(100),
    Eco_Impact_rateing INT
);

CREATE TABLE Orders (
    Order_ID INT PRIMARY KEY,
    Product_ID INT NOT NULL,
    Region VARCHAR(100),
    Units_Sold INT,
    Revenue DECIMAL(10, 2),
    FOREIGN KEY (Product_ID) REFERENCES Products(Product_ID)
);
```

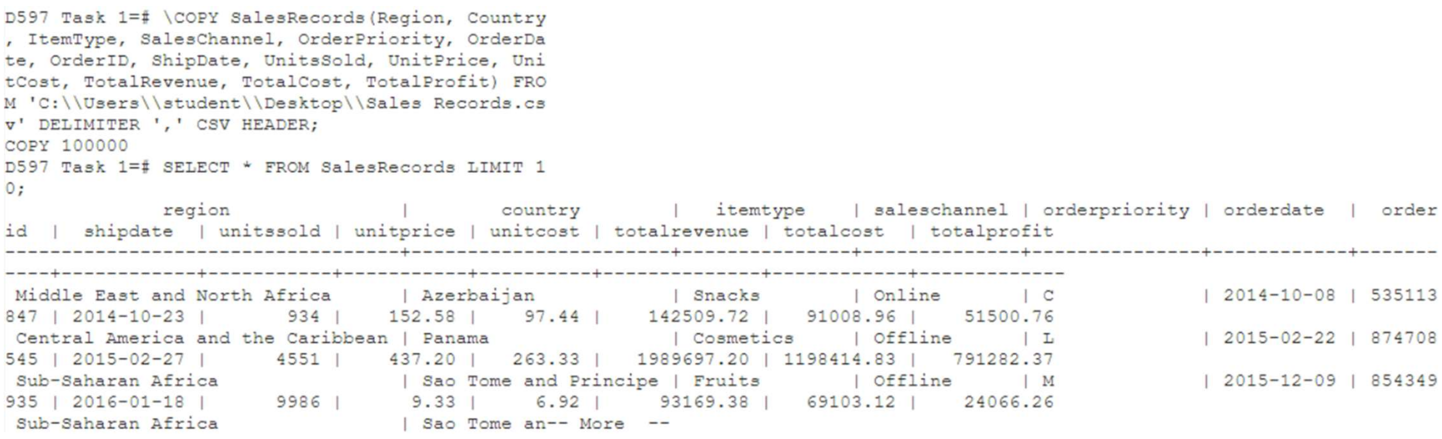


```

CREATE TABLE Reviews (
    Review_ID SERIAL PRIMARY KEY,
    Product_ID INT NOT NULL,
    User_ID INT,
    Rating INT,
    Review_Text TEXT,
    FOREIGN KEY (Product_ID) REFERENCES Products(Product_ID)
);
```



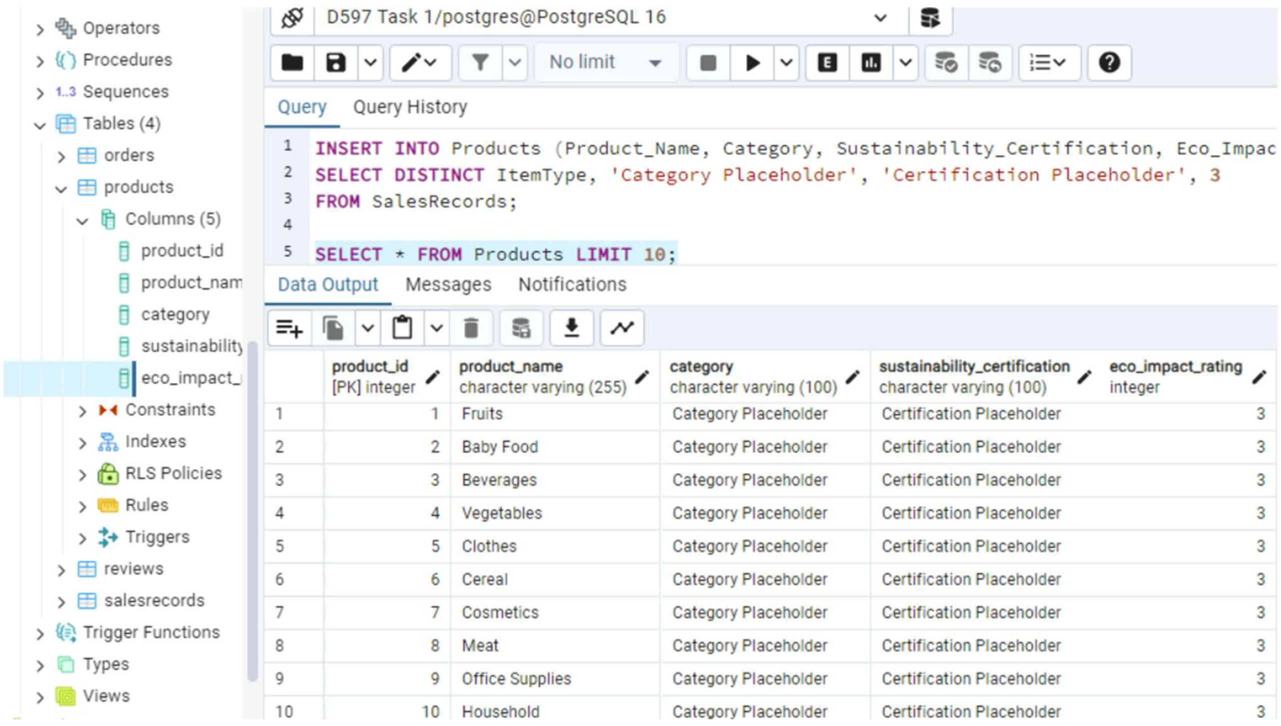
```sql
CREATE TABLE SalesRecords (
    Region VARCHAR(255),
    Country VARCHAR(255),
    ItemType VARCHAR(255),
    SalesChannel VARCHAR(255),
    OrderPriority CHAR(1),
    OrderDate DATE,
    OrderID INT,
    ShipDate DATE,
    UnitsSold INT,
    UnitPrice DECIMAL(10, 2),
    UnitCost DECIMAL(10, 2),
    TotalRevenue DECIMAL(15, 2),
    TotalCost DECIMAL(15, 2),
    TotalProfit DECIMAL(15, 2)
);
```

B. Import Data into the Sales Records Table



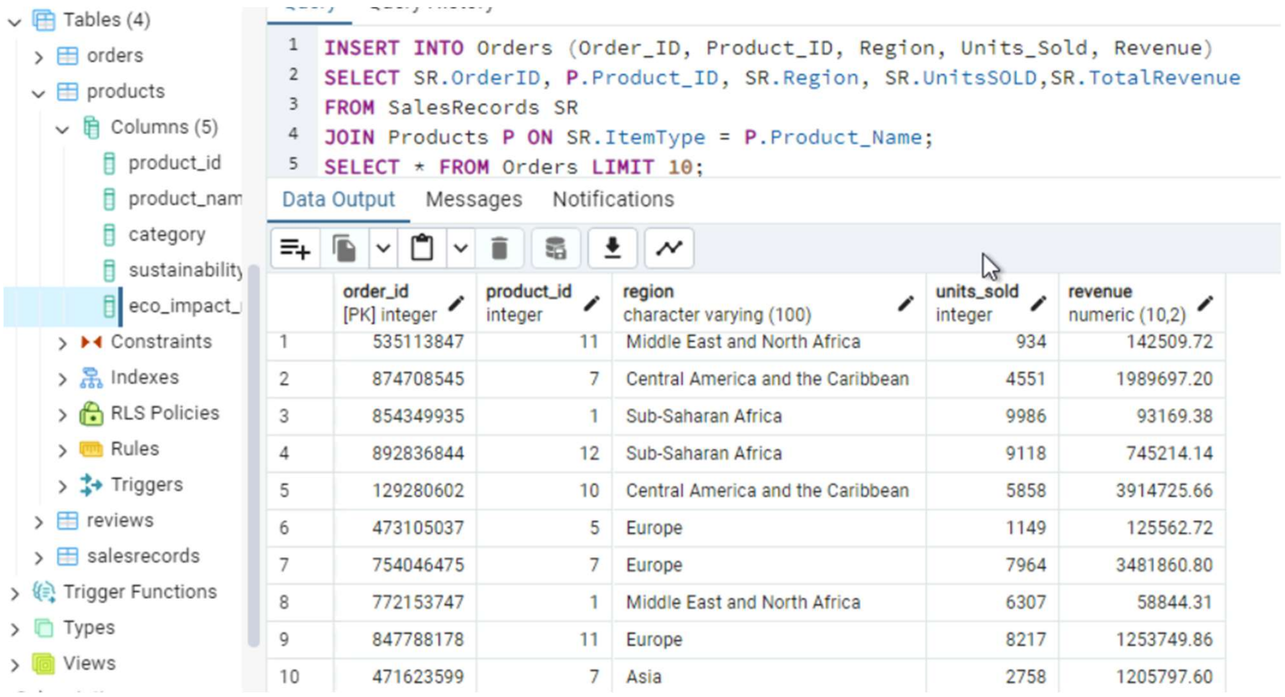
```
D597 Task 1=# \COPY SalesRecords(Region, Country
, ItemType, SalesChannel, OrderPriority, OrderDa
te, OrderID, ShipDate, UnitsSold, UnitPrice, Uni
tCost, TotalRevenue, TotalCost, TotalProfit) FRO
M 'C:\\Users\\student\\Desktop\\Sales Records.cs
v' DELIMITER ',' CSV HEADER;
COPY 100000
D597 Task 1=# SELECT * FROM SalesRecords LIMIT 1
0;
              region              |        country         |  itemtype    | saleschannel | orderpriority | orderdate  |  order
id  |  shipdate  | unitssold | unitprice | unitcost | totalrevenue | totalcost  | totalprofit
----------------------------------+------------------------+--------------+--------------+---------------+------------+-------
----+------------+-----------+-----------+----------+--------------+------------+-------------
 Middle East and North Africa     | Azerbaijan             | Snacks       | Online       | C             | 2014-10-08 | 535113
847 | 2014-10-23 |       934 |    152.58 |    97.44 |    142509.72 |   91008.96 |    51500.76
 Central America and the Caribbean | Panama                | Cosmetics    | Offline      | L             | 2015-02-22 | 874708
545 | 2015-02-27 |      4551 |    437.20 |   263.33 |   1989697.20 | 1198414.83 |   791282.37
 Sub-Saharan Africa               | Sao Tome and Principe  | Fruits       | Offline      | M             | 2015-12-09 | 854349
935 | 2016-01-18 |      9986 |      9.33 |     6.92 |     93169.38 |   69103.12 |    24066.26
 Sub-Saharan Africa               | Sao Tome an-- More  --
```

C. Populate Products, Orders, and Reviews Tables



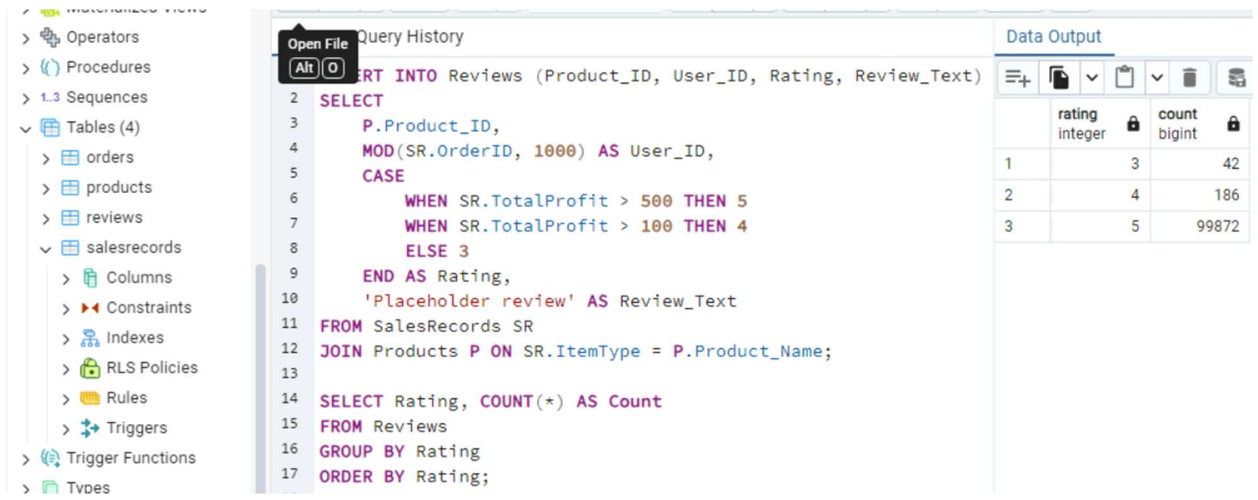
```sql
INSERT INTO Products (Product_Name, Category, Sustainability_Certification, Eco_Impac
SELECT DISTINCT ItemType, 'Category Placeholder', 'Certification Placeholder', 3
FROM SalesRecords;

SELECT * FROM Products LIMIT 10;
```

	product_id [PK] integer	product_name character varying (255)	category character varying (100)	sustainability_certification character varying (100)	eco_impact_rating integer
1	1	Fruits	Category Placeholder	Certification Placeholder	3
2	2	Baby Food	Category Placeholder	Certification Placeholder	3
3	3	Beverages	Category Placeholder	Certification Placeholder	3
4	4	Vegetables	Category Placeholder	Certification Placeholder	3
5	5	Clothes	Category Placeholder	Certification Placeholder	3
6	6	Cereal	Category Placeholder	Certification Placeholder	3
7	7	Cosmetics	Category Placeholder	Certification Placeholder	3
8	8	Meat	Category Placeholder	Certification Placeholder	3
9	9	Office Supplies	Category Placeholder	Certification Placeholder	3
10	10	Household	Category Placeholder	Certification Placeholder	3



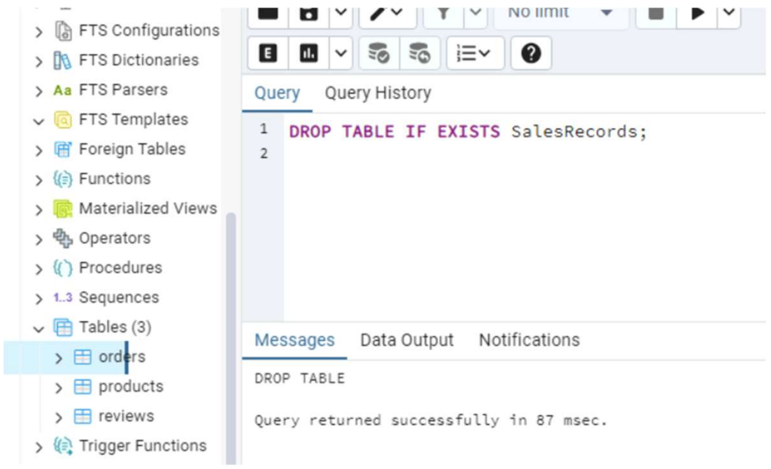
```
INSERT INTO Orders (Order_ID, Product_ID, Region, Units_Sold, Revenue)
SELECT SR.OrderID, P.Product_ID, SR.Region, SR.UnitsSOLD,SR.TotalRevenue
FROM SalesRecords SR
JOIN Products P ON SR.ItemType = P.Product_Name;
SELECT * FROM Orders LIMIT 10;
```

	order_id [PK] integer	product_id integer	region character varying (100)	units_sold integer	revenue numeric (10,2)
1	535113847	11	Middle East and North Africa	934	142509.72
2	874708545	7	Central America and the Caribbean	4551	1989697.20
3	854349935	1	Sub-Saharan Africa	9986	93169.38
4	892836844	12	Sub-Saharan Africa	9118	745214.14
5	129280602	10	Central America and the Caribbean	5858	3914725.66
6	473105037	5	Europe	1149	125562.72
7	754046475	7	Europe	7964	3481860.80
8	772153747	1	Middle East and North Africa	6307	58844.31
9	847788178	11	Europe	8217	1253749.86
10	471623599	7	Asia	2758	1205797.60



```
RT INTO Reviews (Product_ID, User_ID, Rating, Review_Text)
SELECT
    P.Product_ID,
    MOD(SR.OrderID, 1000) AS User_ID,
    CASE
        WHEN SR.TotalProfit > 500 THEN 5
        WHEN SR.TotalProfit > 100 THEN 4
        ELSE 3
    END AS Rating,
    'Placeholder review' AS Review_Text
FROM SalesRecords SR
JOIN Products P ON SR.ItemType = P.Product_Name;

SELECT Rating, COUNT(*) AS Count
FROM Reviews
GROUP BY Rating
ORDER BY Rating;
```

	rating integer	count bigint
1	3	42
2	4	186
3	5	99872

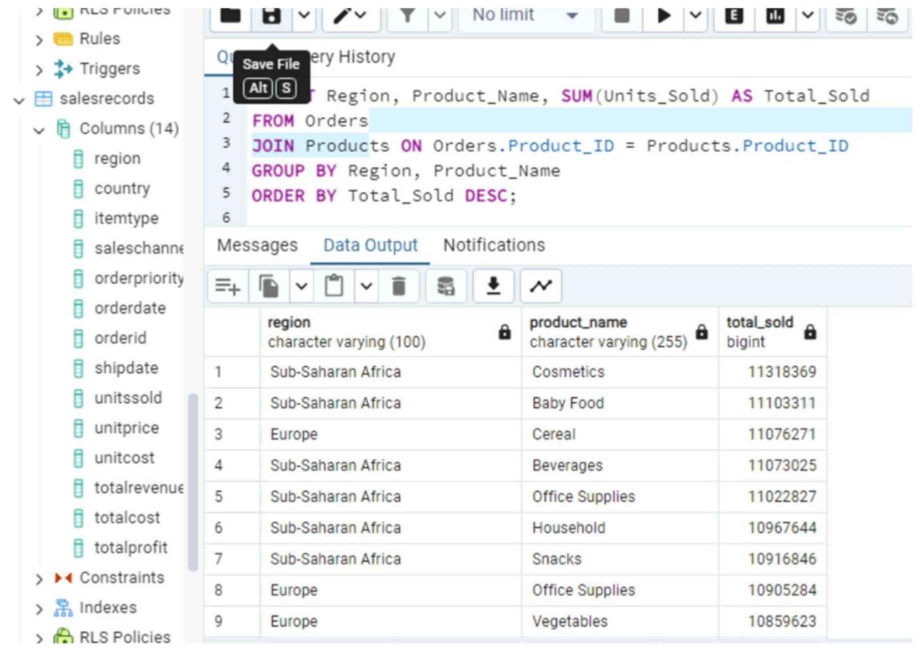


```
DROP TABLE IF EXISTS SalesRecords;
```

```
DROP TABLE

Query returned successfully in 87 msec.
```

F3. Write Queries for Business Insights

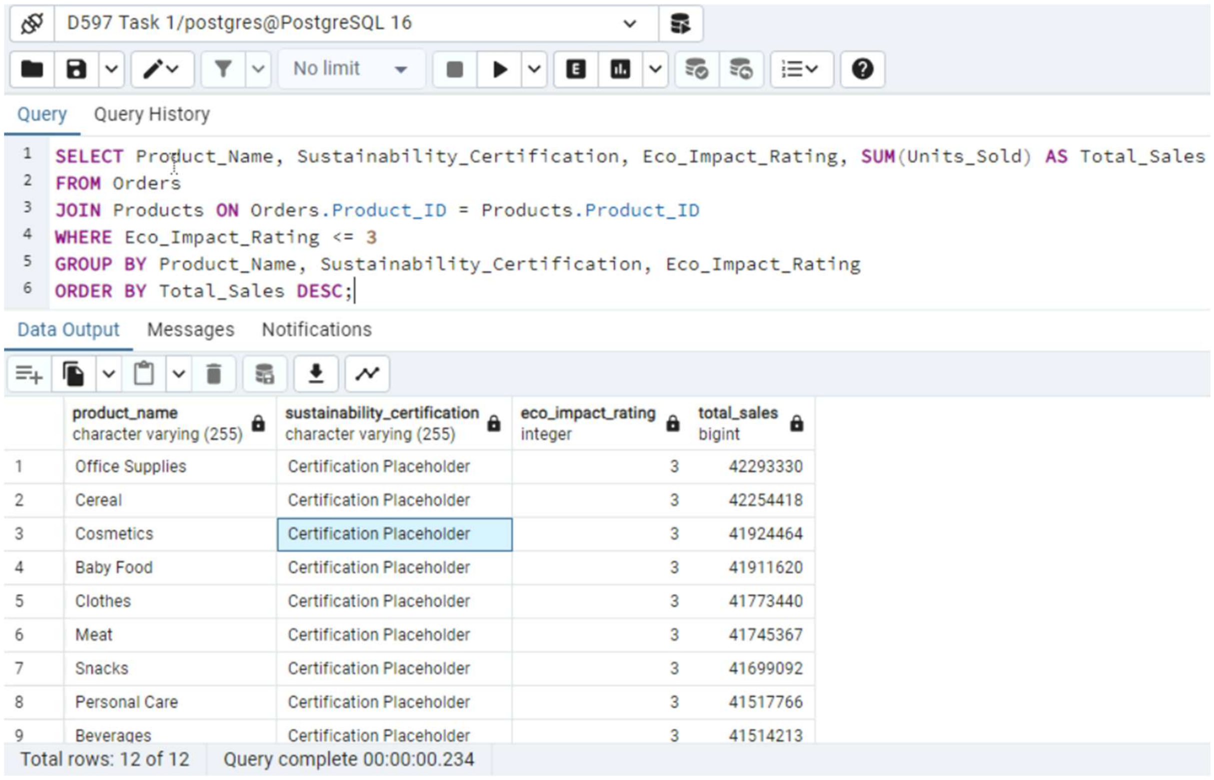
Query 1: Top-Selling Products by Region



```
SELECT Region, Product_Name, SUM(Units_Sold) AS Total_Sold
FROM Orders
JOIN Products ON Orders.Product_ID = Products.Product_ID
GROUP BY Region, Product_Name
ORDER BY Total_Sold DESC;
```

	region character varying (100)	product_name character varying (255)	total_sold bigint
1	Sub-Saharan Africa	Cosmetics	11318369
2	Sub-Saharan Africa	Baby Food	11103311
3	Europe	Cereal	11076271
4	Sub-Saharan Africa	Beverages	11073025
5	Sub-Saharan Africa	Office Supplies	11022827
6	Sub-Saharan Africa	Household	10967644
7	Sub-Saharan Africa	Snacks	10916846
8	Europe	Office Supplies	10905284
9	Europe	Vegetables	10859623

Query 2: Eco-Friendly Products with High Sales



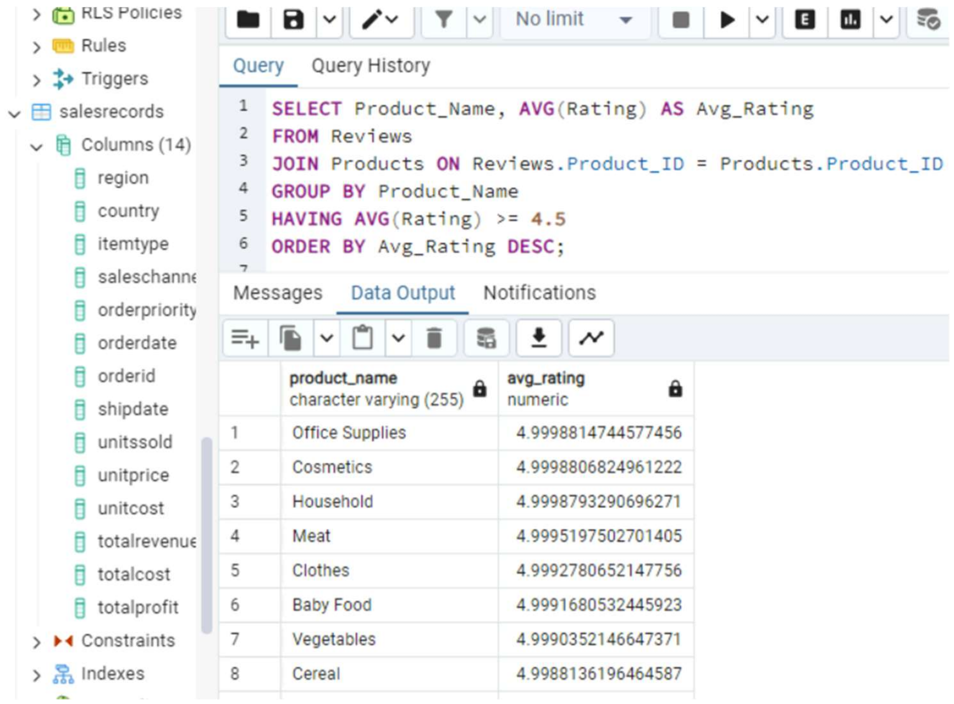
D597 Task 1/postgres@PostgreSQL 16

```
SELECT Product_Name, Sustainability_Certification, Eco_Impact_Rating, SUM(Units_Sold) AS Total_Sales
FROM Orders
JOIN Products ON Orders.Product_ID = Products.Product_ID
WHERE Eco_Impact_Rating <= 3
GROUP BY Product_Name, Sustainability_Certification, Eco_Impact_Rating
ORDER BY Total_Sales DESC;
```

	product_name character varying (255)	sustainability_certification character varying (255)	eco_impact_rating integer	total_sales bigint
1	Office Supplies	Certification Placeholder	3	42293330
2	Cereal	Certification Placeholder	3	42254418
3	Cosmetics	Certification Placeholder	3	41924464
4	Baby Food	Certification Placeholder	3	41911620
5	Clothes	Certification Placeholder	3	41773440
6	Meat	Certification Placeholder	3	41745367
7	Snacks	Certification Placeholder	3	41699092
8	Personal Care	Certification Placeholder	3	41517766
9	Beverages	Certification Placeholder	3	41514213

Total rows: 12 of 12 Query complete 00:00:00.234

Query 3: Top-Rated Products



```
SELECT Product_Name, AVG(Rating) AS Avg_Rating
FROM Reviews
JOIN Products ON Reviews.Product_ID = Products.Product_ID
GROUP BY Product_Name
HAVING AVG(Rating) >= 4.5
ORDER BY Avg_Rating DESC;
```

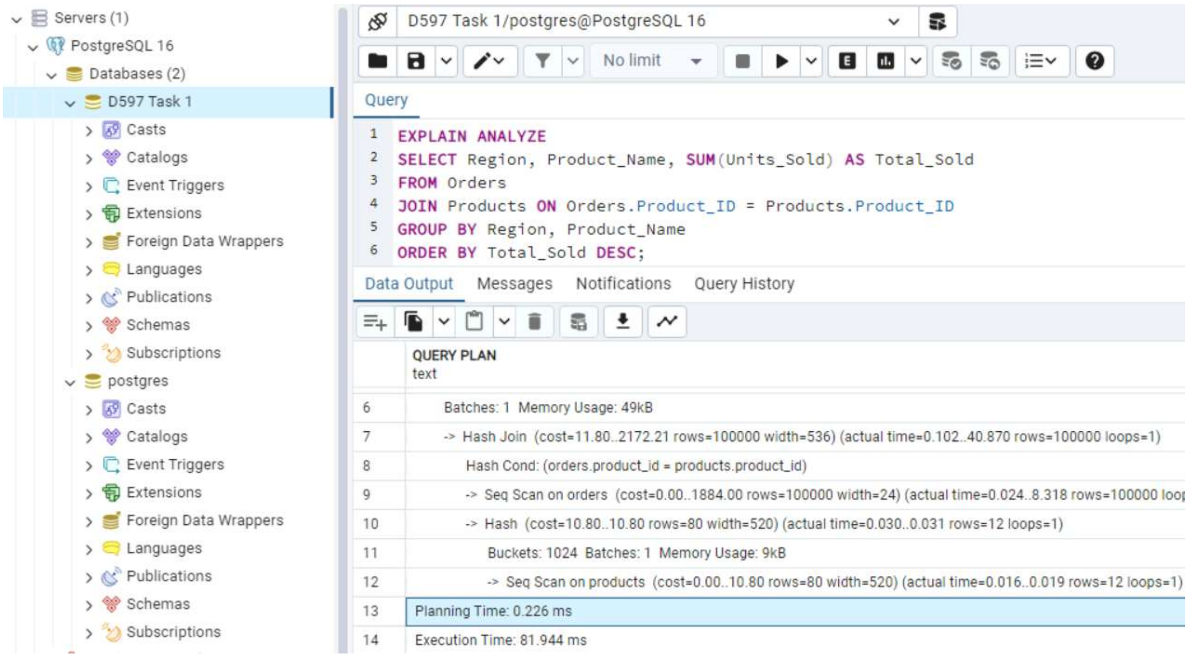
	product_name character varying (255)	avg_rating numeric
1	Office Supplies	4.9998814744577456
2	Cosmetics	4.9998806824961222
3	Household	4.9998793290696271
4	Meat	4.9995197502701405
5	Clothes	4.9992780652147756
6	Baby Food	4.9991680532445923
7	Vegetables	4.9990352146647371
8	Cereal	4.9988136196464587

F4.) Optimization of Scripts and Data Validation

a. Optimization Scripts

Pre-indexing

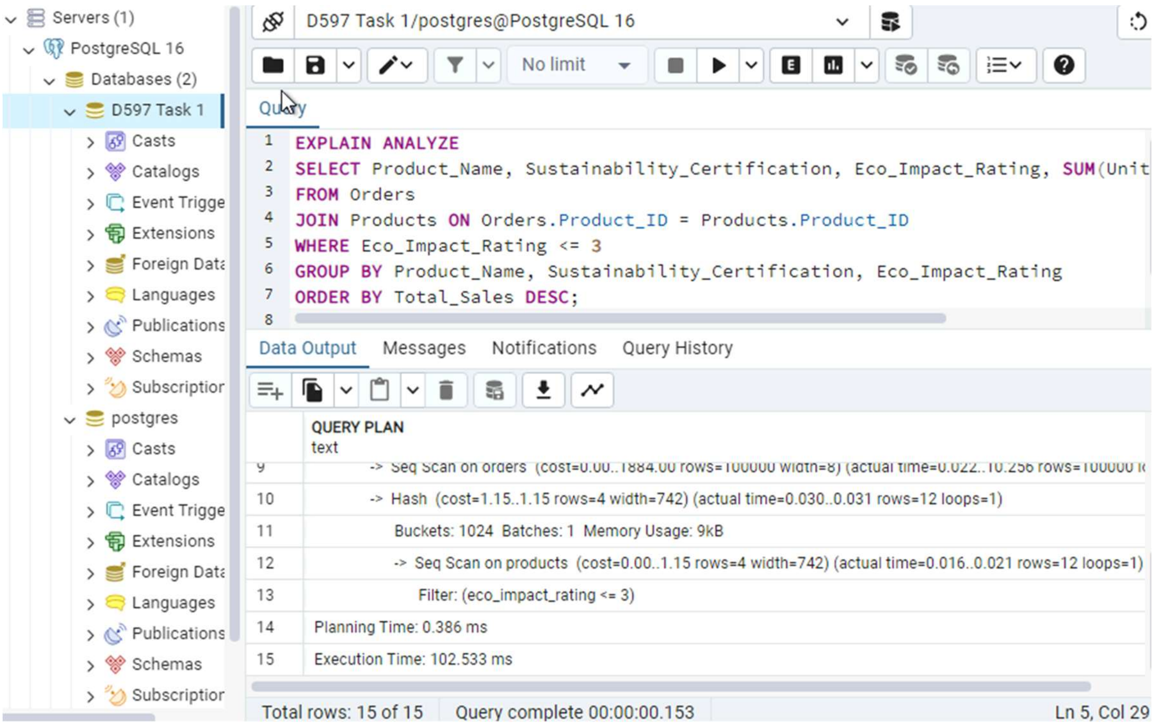
Query 1



```sql
EXPLAIN ANALYZE
SELECT Region, Product_Name, SUM(Units_Sold) AS Total_Sold
FROM Orders
JOIN Products ON Orders.Product_ID = Products.Product_ID
GROUP BY Region, Product_Name
ORDER BY Total_Sold DESC;
```

```
QUERY PLAN
text
6        Batches: 1  Memory Usage: 49kB
7        -> Hash Join  (cost=11.80..2172.21 rows=100000 width=536) (actual time=0.102..40.870 rows=100000 loops=1)
8              Hash Cond: (orders.product_id = products.product_id)
9              -> Seq Scan on orders  (cost=0.00..1884.00 rows=100000 width=24) (actual time=0.024..8.318 rows=100000 loop
10             -> Hash  (cost=10.80..10.80 rows=80 width=520) (actual time=0.030..0.031 rows=12 loops=1)
11                   Buckets: 1024  Batches: 1  Memory Usage: 9kB
12                   -> Seq Scan on products  (cost=0.00..10.80 rows=80 width=520) (actual time=0.016..0.019 rows=12 loops=1)
13   Planning Time: 0.226 ms
14   Execution Time: 81.944 ms
```

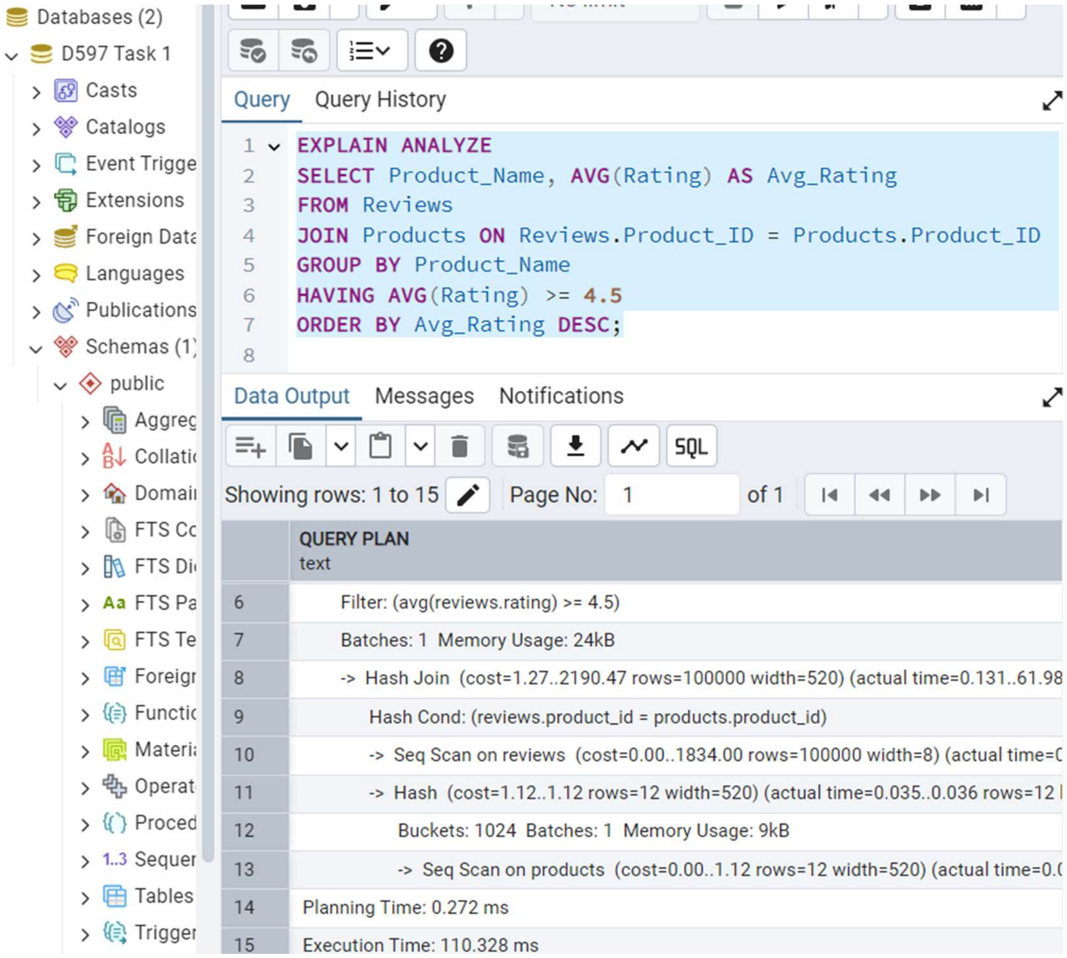
Query 2



```sql
EXPLAIN ANALYZE
SELECT Product_Name, Sustainability_Certification, Eco_Impact_Rating, SUM(Unit
FROM Orders
JOIN Products ON Orders.Product_ID = Products.Product_ID
WHERE Eco_Impact_Rating <= 3
GROUP BY Product_Name, Sustainability_Certification, Eco_Impact_Rating
ORDER BY Total_Sales DESC;
```

```
QUERY PLAN
text
9              -> Seq Scan on orders  (cost=0.00..1884.00 rows=100000 width=8) (actual time=0.022..10.256 rows=100000 l
10             -> Hash  (cost=1.15..1.15 rows=4 width=742) (actual time=0.030..0.031 rows=12 loops=1)
11                   Buckets: 1024  Batches: 1  Memory Usage: 9kB
12                   -> Seq Scan on products  (cost=0.00..1.15 rows=4 width=742) (actual time=0.016..0.021 rows=12 loops=1)
13                         Filter: (eco_impact_rating <= 3)
14   Planning Time: 0.386 ms
15   Execution Time: 102.533 ms

Total rows: 15 of 15    Query complete 00:00:00.153    Ln 5, Col 29
```

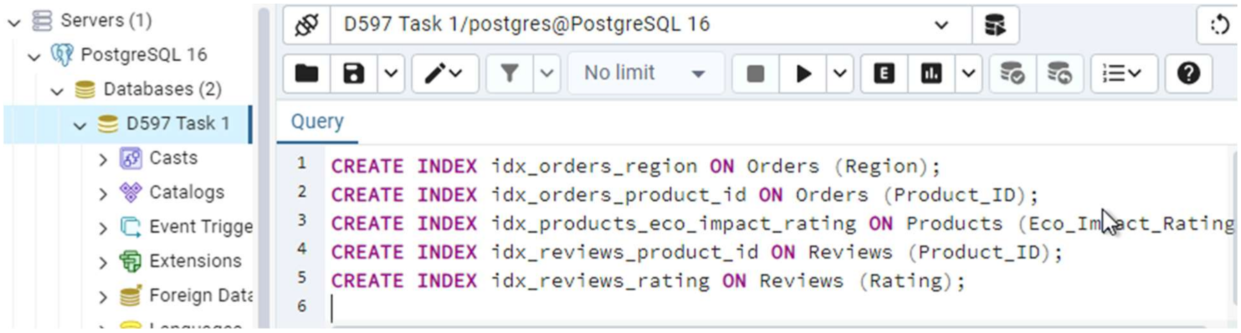
Query 3



```sql
EXPLAIN ANALYZE
SELECT Product_Name, AVG(Rating) AS Avg_Rating
FROM Reviews
JOIN Products ON Reviews.Product_ID = Products.Product_ID
GROUP BY Product_Name
HAVING AVG(Rating) >= 4.5
ORDER BY Avg_Rating DESC;
```

```
QUERY PLAN
text
      Filter: (avg(reviews.rating) >= 4.5)
      Batches: 1  Memory Usage: 24kB
      ->  Hash Join  (cost=1.27..2190.47 rows=100000 width=520) (actual time=0.131..61.98
            Hash Cond: (reviews.product_id = products.product_id)
            ->  Seq Scan on reviews  (cost=0.00..1834.00 rows=100000 width=8) (actual time=0
            ->  Hash  (cost=1.12..1.12 rows=12 width=520) (actual time=0.035..0.036 rows=12 l
                  Buckets: 1024  Batches: 1  Memory Usage: 9kB
                  ->  Seq Scan on products  (cost=0.00..1.12 rows=12 width=520) (actual time=0.0
Planning Time: 0.272 ms
Execution Time: 110.328 ms
```

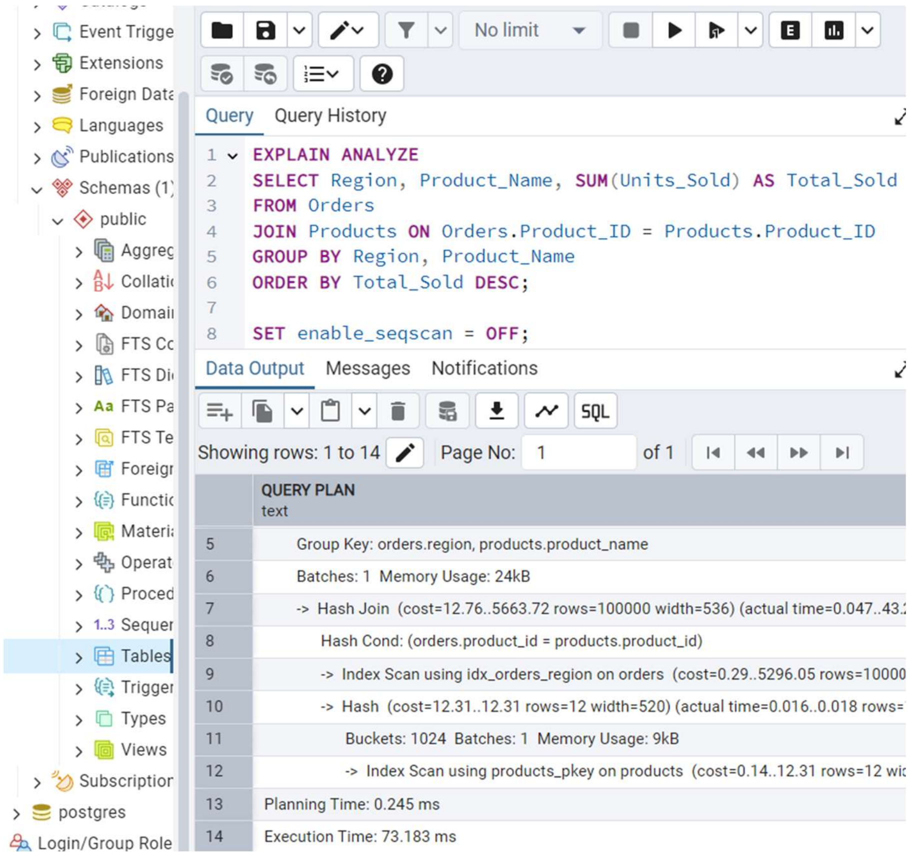
Indexing



```sql
CREATE INDEX idx_orders_region ON Orders (Region);
CREATE INDEX idx_orders_product_id ON Orders (Product_ID);
CREATE INDEX idx_products_eco_impact_rating ON Products (Eco_Impact_Rating
CREATE INDEX idx_reviews_product_id ON Reviews (Product_ID);
CREATE INDEX idx_reviews_rating ON Reviews (Rating);
```

Post-indexing

Query 1

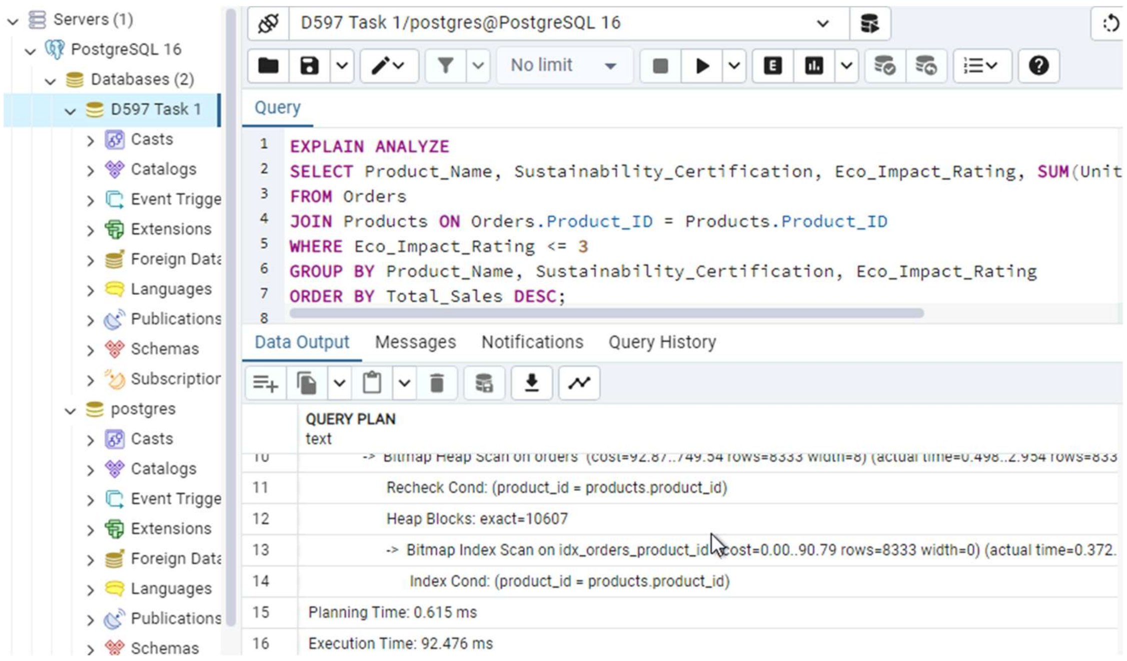


```
EXPLAIN ANALYZE
SELECT Region, Product_Name, SUM(Units_Sold) AS Total_Sold
FROM Orders
JOIN Products ON Orders.Product_ID = Products.Product_ID
GROUP BY Region, Product_Name
ORDER BY Total_Sold DESC;

SET enable_seqscan = OFF;
```

Showing rows: 1 to 14   Page No: 1 of 1

```
QUERY PLAN
text
5         Group Key: orders.region, products.product_name
6         Batches: 1  Memory Usage: 24kB
7         ->  Hash Join  (cost=12.76..5663.72 rows=100000 width=536) (actual time=0.047..43.
8               Hash Cond: (orders.product_id = products.product_id)
9               ->  Index Scan using idx_orders_region on orders  (cost=0.29..5296.05 rows=10000
10              ->  Hash  (cost=12.31..12.31 rows=12 width=520) (actual time=0.016..0.018 rows=
11                    Buckets: 1024  Batches: 1  Memory Usage: 9kB
12                    ->  Index Scan using products_pkey on products  (cost=0.14..12.31 rows=12 wi
13  Planning Time: 0.245 ms
14  Execution Time: 73.183 ms
```

Query 2

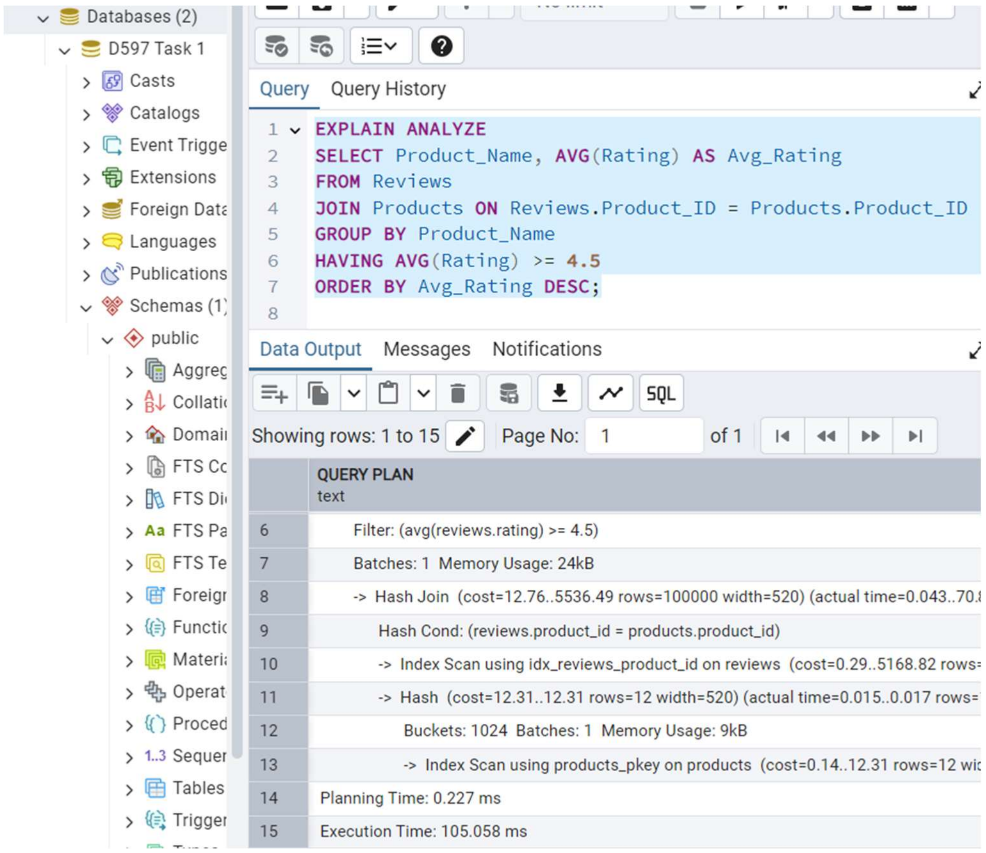


D597 Task 1/postgres@PostgreSQL 16

```
EXPLAIN ANALYZE
SELECT Product_Name, Sustainability_Certification, Eco_Impact_Rating, SUM(Unit
FROM Orders
JOIN Products ON Orders.Product_ID = Products.Product_ID
WHERE Eco_Impact_Rating <= 3
GROUP BY Product_Name, Sustainability_Certification, Eco_Impact_Rating
ORDER BY Total_Sales DESC;
```

```
QUERY PLAN
text
10          ->  Bitmap Heap Scan on orders  (cost=92.87..749.54 rows=8333 width=8) (actual time=0.498..2.954 rows=833
11                Recheck Cond: (product_id = products.product_id)
12                Heap Blocks: exact=10607
13                ->  Bitmap Index Scan on idx_orders_product_id  (cost=0.00..90.79 rows=8333 width=0) (actual time=0.372.
14                      Index Cond: (product_id = products.product_id)
15  Planning Time: 0.615 ms
16  Execution Time: 92.476 ms
```

Query 3

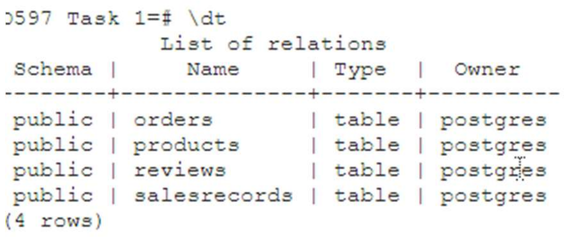


```sql
EXPLAIN ANALYZE
SELECT Product_Name, AVG(Rating) AS Avg_Rating
FROM Reviews
JOIN Products ON Reviews.Product_ID = Products.Product_ID
GROUP BY Product_Name
HAVING AVG(Rating) >= 4.5
ORDER BY Avg_Rating DESC;
```

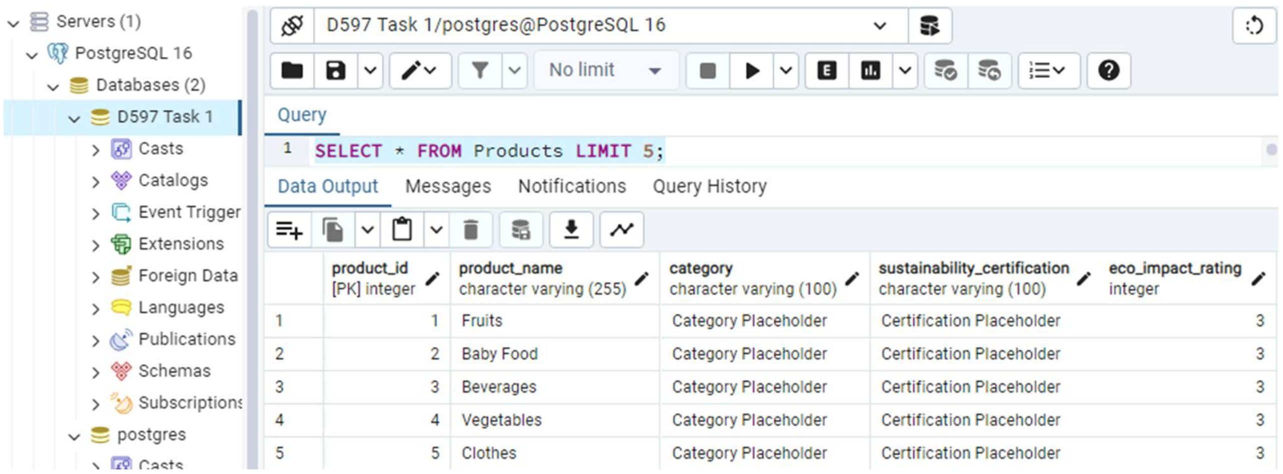
Showing rows: 1 to 15 Page No: 1 of 1

```
QUERY PLAN
text
6        Filter: (avg(reviews.rating) >= 4.5)
7        Batches: 1  Memory Usage: 24kB
8        -> Hash Join  (cost=12.76..5536.49 rows=100000 width=520) (actual time=0.043..70.
9              Hash Cond: (reviews.product_id = products.product_id)
10             -> Index Scan using idx_reviews_product_id on reviews  (cost=0.29..5168.82 rows=
11             -> Hash  (cost=12.31..12.31 rows=12 width=520) (actual time=0.015..0.017 rows=
12                   Buckets: 1024  Batches: 1  Memory Usage: 9kB
13                   -> Index Scan using products_pkey on products  (cost=0.14..12.31 rows=12 wi
14 Planning Time: 0.227 ms
15 Execution Time: 105.058 ms
```

b. Final Data Validation



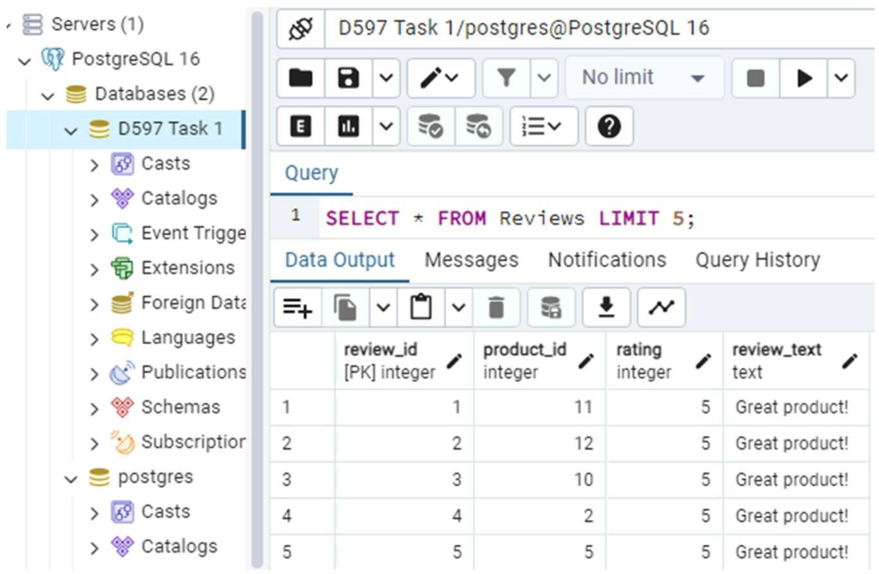
```
D597 Task 1=# \dt
            List of relations
 Schema |     Name      | Type  |  Owner
--------+---------------+-------+----------
 public | orders        | table | postgres
 public | products      | table | postgres
 public | reviews       | table | postgres
 public | salesrecords  | table | postgres
(4 rows)
```



D597 Task 1/postgres@PostgreSQL 16

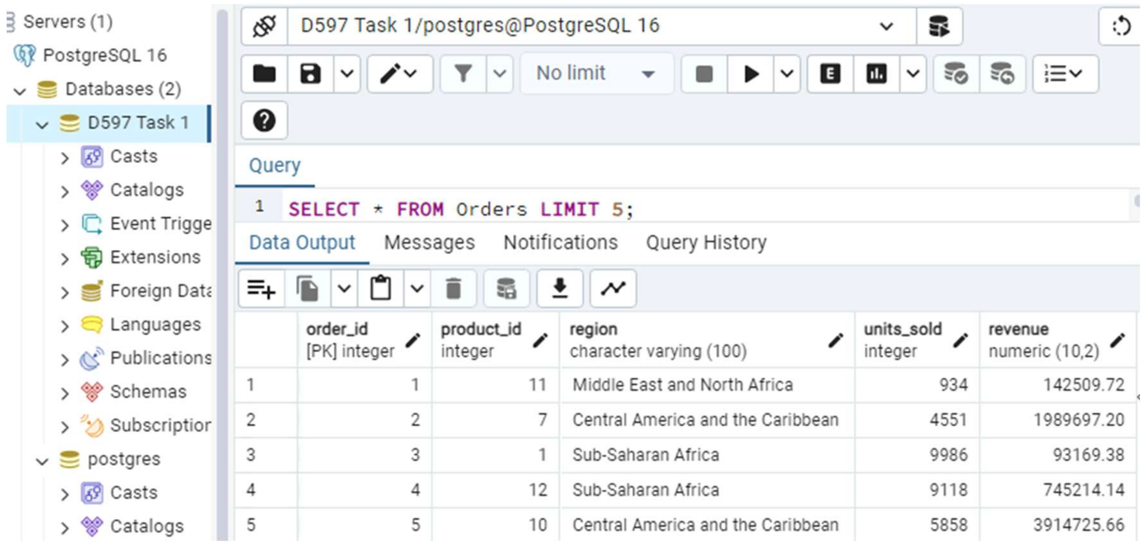
```sql
SELECT * FROM Products LIMIT 5;
```

	product_id [PK] integer	product_name character varying (255)	category character varying (100)	sustainability_certification character varying (100)	eco_impact_rating integer
1	1	Fruits	Category Placeholder	Certification Placeholder	3
2	2	Baby Food	Category Placeholder	Certification Placeholder	3
3	3	Beverages	Category Placeholder	Certification Placeholder	3
4	4	Vegetables	Category Placeholder	Certification Placeholder	3
5	5	Clothes	Category Placeholder	Certification Placeholder	3



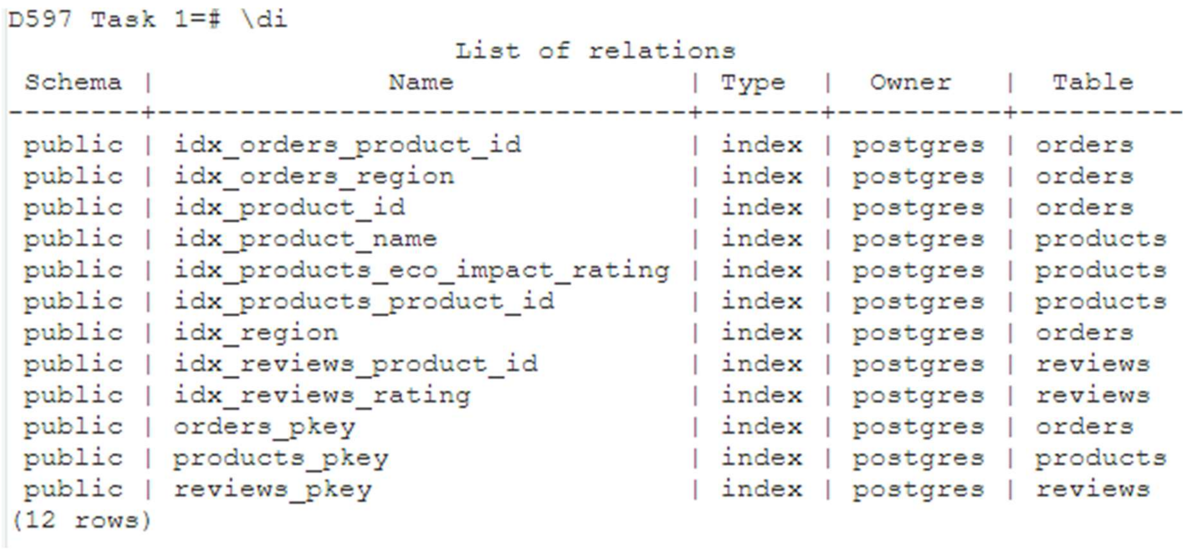
```sql
SELECT * FROM Reviews LIMIT 5;
```

	review_id [PK] integer	product_id integer	rating integer	review_text text
1	1	11	5	Great product!
2	2	12	5	Great product!
3	3	10	5	Great product!
4	4	2	5	Great product!
5	5	5	5	Great product!



```sql
SELECT * FROM Orders LIMIT 5;
```

	order_id [PK] integer	product_id integer	region character varying (100)	units_sold integer	revenue numeric (10,2)
1	1	11	Middle East and North Africa	934	142509.72
2	2	7	Central America and the Caribbean	4551	1989697.20
3	3	1	Sub-Saharan Africa	9986	93169.38
4	4	12	Sub-Saharan Africa	9118	745214.14
5	5	10	Central America and the Caribbean	5858	3914725.66



```
D597 Task 1=# \di
                         List of relations
 Schema |              Name              | Type  |  Owner   |  Table
--------+--------------------------------+-------+----------+----------
 public | idx_orders_product_id          | index | postgres | orders
 public | idx_orders_region              | index | postgres | orders
 public | idx_product_id                 | index | postgres | orders
 public | idx_product_name               | index | postgres | products
 public | idx_products_eco_impact_rating | index | postgres | products
 public | idx_products_product_id        | index | postgres | products
 public | idx_region                     | index | postgres | orders
 public | idx_reviews_product_id         | index | postgres | reviews
 public | idx_reviews_rating             | index | postgres | reviews
 public | orders_pkey                    | index | postgres | orders
 public | products_pkey                  | index | postgres | products
 public | reviews_pkey                   | index | postgres | reviews
(12 rows)
```

References

Elmasri, R., & Navathe, S. B. (2016). *Fundamentals of Database Systems*. Pearson Education.

Pang, B., & Lee, L. (2008). *Opinion Mining and Sentiment Analysis*. Foundations and Trends in Information Retrieval.

PostgreSQL Global Development Group. (n.d.). *PostgreSQL Documentation*. Retrieved from https://www.postgresql.org/docs/

Stonebraker, M., & Hellerstein, J. M. (2005). *What Goes Around Comes Around*. Communications of the ACM, 48(5), 67–72.

United Nations. (2015). *Transforming our world: The 2030 agenda for sustainable development*. Retrieved from https://sdgs.un.org/2030agenda