Business Problem

HealthFit Innovations is facing significant challenges with its existing database infrastructure as it seeks to expand the HealthTrack platform. The current system cannot manage the diverse and dynamic nature of the health-related data generated by wearable devices, electronic health records (EHRs), and other sources. Specifically, the existing database management system (DBMS) has become a bottleneck due to its lack of scalability, leading to issues such as:

1. **Performance Bottlenecks**: The massive influx of real-time data from users, including metrics like heart rate, sleep patterns, and blood glucose levels, overwhelms the system, resulting in slow query processing and delayed insights.
2. **Data Silos**: The inability to integrate data from multiple sources efficiently creates fragmented and isolated datasets, hindering comprehensive analysis and personalized health recommendations.
3. **Integration and Analysis Challenges**: HealthTrack aims to provide personalized health insights and predictive analytics, but the current system's rigidity limits the effective organization, integration, and analysis of heterogeneous data types.

These challenges affect user experience and limit HealthFit's ability to scale operations and deliver innovative, data-driven healthcare solutions. A robust, scalable, and flexible database solution is critical to address these limitations and support HealthTrack’s goals of real-time monitoring, personalized insights, and predictive analytics.

Proposed Data Structure

A relational database model will be implemented to address the challenges faced by HealthFit Innovations. This model organizes the diverse and dynamic data generated by the HealthTrack platform into well-structured tables with transparent relationships, ensuring scalability, integration, and efficient analysis. The following data structure is proposed:

1. **Users Table**
    - Stores demographic and identification information for individual users.
    - **Attributes**:
        - `UserID` (Primary Key): Unique identifier for each user.
        - `FirstName`, `LastName`
        - `DateOfBirth`
        - `Gender`
        - `Email`
        - `PhoneNumber`
        - `SignupDate`
2. **Devices Table**
    - Captures details about the wearable and medical devices associated with users.
    - **Attributes**:
        - `DeviceID` (Primary Key): Unique identifier for each device.

        - `UserID` (Foreign Key): Links devices to users.
        - `DeviceType`: Type of device (e.g., fitness tracker, smartwatch, glucose monitor).
        - `DeviceModel`
        - `RegistrationDate`
3. **HealthMetrics Table**
    - Logs health-related data collected from devices.
    - **Attributes**:
        - `MetricID` (Primary Key): Unique identifier for each recorded metric.
        - `UserID` (Foreign Key): Links metrics to users.
        - `DeviceID` (Foreign Key): Links metrics to the originating device.
        - `MetricType`: Type of metric (e.g., heart rate, sleep, steps, blood glucose).
        - `MetricValue`
        - `Timestamp`: Date and time the metric was recorded.
4. **ElectronicHealthRecords (EHR) Table**
    - Stores data from users' electronic health records.
    - **Attributes**:
        - `RecordID` (Primary Key): Unique identifier for each EHR.
        - `UserID` (Foreign Key): Links records to users.
        - `RecordType`: Type of record (e.g., diagnosis, lab result, medication).
        - `RecordDetails`
        - `RecordDate`
5. **Recommendations Table**
    - Maintains personalized health recommendations for users.
    - **Attributes**:
        - `RecommendationID` (Primary Key): Unique identifier for each recommendation.
        - `UserID` (Foreign Key): Links recommendations to users.
        - `RecommendationDetails`
        - `GeneratedDate`
6. **ActivityLog Table**
    - Tracks user interactions and system-generated events for auditing and insights.
    - **Attributes**:
        - `LogID` (Primary Key): Unique identifier for each activity log.
        - `UserID` (Foreign Key): Links logs to users.
        - `ActivityType`: Type of activity (e.g., login, data sync, profile update).
        - `Timestamp`
7. **Relationships Between Tables**
    - **Users** ↔ **Devices**: One user can own multiple devices.
    - **Users** ↔ **HealthMetrics**: One user can have many health metrics logged.
    - **Users** ↔ **EHR**: One user can have multiple electronic health records.

- **Users** ↔ **Recommendations**: One user can have multiple personalized recommendations.
- **Users** ↔ **ActivityLog**: One user can have many activity logs.

Justification for a Database Solution

A relational database solution addresses HealthFit Innovations' business problem effectively by offering the following advantages:

1. **Scalability and Performance**
   - The proposed database is designed to scale horizontally or vertically as HealthTrack's user base grows and the volume of data increases.
   - Relational databases can leverage indexing, partitioning, and optimized query processing to handle real-time data ingestion and analytics, minimizing performance bottlenecks.
2. **Data Integration**
   - Relational databases are well-suited for consolidating diverse data sources such as wearable devices, electronic health records (EHRs), and user-reported outcomes.
   - By defining transparent relationships between tables, the system eliminates data silos and supports seamless data integration.
3. **Data Organization and Retrieval**
   - The structured format of relational databases ensures efficient data storage and retrieval.
   - Queries can be written to analyze data across multiple dimensions, such as linking health metrics to user profiles or generating health recommendations based on activity logs.
4. **Flexibility for Dynamic Data**
   - Health-related data is highly diverse, from continuous metrics like heart rate to categorical data like diagnosis codes. The relational model supports such diversity through table normalization and relationships.
   - Adding tables or columns to the database can integrate new data types or sources without disrupting the existing schema.
5. **Real-Time Insights and Analytics**
   - The database supports real-time monitoring and predictive analytics by enabling efficient storage and processing of timestamped health metrics and logs.
   - HealthTrack's personalized recommendations rely on analyzing large datasets quickly, which a well-designed database can handle efficiently.
6. **Data Security and Compliance**
   - Relational databases offer robust security features such as role-based access control, encryption, and audit trails.
   - These features are essential for handling sensitive healthcare data and ensuring compliance with regulations like HIPAA.
7. **Improved User Experience**

    - Faster query response times and integrated data insights enhance the user experience for individuals and healthcare providers.
    - A reliable database ensures that HealthTrack can deliver timely and accurate health recommendations, fostering user trust and engagement.
8. **Support for Future Growth**
    - A relational database provides a solid foundation for integrating advanced functionalities, such as AI-driven analytics and machine learning models, to further enhance HealthTrack's capabilities.

By implementing a relational database solution, HealthFit can overcome the limitations of its current infrastructure, efficiently manage the growing complexity of health-related data, and deliver on its promise of innovative, data-driven healthcare solutions.

Using Business Data Within the Database Solution

The business data collected by HealthTrack will be used in the following ways within the relational database solution to enhance functionality, improve user experience, and drive business outcomes:

1. **User Profile Management**
    - **Data Used**: User demographics and contact information are stored in the `Users` table.
    - **Usage**:
        - Allows the platform to identify and manage individual users.
        - Enables personalized communication, such as sending health alerts or recommendations.
        - Supports compliance with data privacy regulations by associating user consent preferences with their profiles.
2. **Device Integration and Monitoring**
    - **Data Used**: Device details are stored in the `Devices` table.
    - **Usage**:
        - Tracks which devices are linked to each user.
        - Ensures compatibility and data synchronization between various wearable devices and the platform.
        - Facilitates troubleshooting and support for device-related issues.
3. **Health Metrics Analysis**
    - **Data Used**: Wearable-generated data from the `HealthMetrics` table.
    - **Usage**:
        - Enables real-time analysis of user health data, such as identifying irregular heart rates or sleep disruptions.
        - Supports the generation of insights and trends, such as weekly activity summaries or health improvement suggestions.
        - Feeds data into predictive models for identifying potential health risks.
4. **Personalized Health Recommendations**
    - **Data Used**: Aggregated insights from `HealthMetrics`, EHR tables, and user preferences.

    - **Usage**:
        - The `Recommendations` table is populated with tailored advice, such as reminders to increase activity levels or dietary changes.
        - Recommendations are generated based on combined metrics, such as linking a sedentary week with dietary habits or medical conditions.
        - Provides actionable, easy-to-understand suggestions to improve user health outcomes.
5. **Medical History Management**
    - **Data Used**: EHR data is stored in the `electronic health records` table.
    - **Usage**:
        - Centralizes medical records for easier access by users and healthcare providers.
        - Facilitates longitudinal studies of a user's health progress over time.
        - Enables advanced analyses, such as correlating device data with specific diagnoses or treatments.
6. **Audit Trails and User Engagement**
    - **Data Used**: Logs stored in the `ActivityLog` table.
    - **Usage**:
        - Tracks user interactions with the platform, such as logins, data uploads, and profile updates.
        - Supports system auditing to ensure data security and adherence to privacy policies.
        - Provides insights into user behavior patterns, such as active vs. inactive users, to improve engagement strategies.
7. **Platform Optimization and Reporting**
    - **Data Used**: Aggregated and relational data across all tables.
    - **Usage**:
        - Enables real-time reporting for platform administrators, such as system usage, device performance, and health trends.
        - Data can be used to optimize the platform, such as improving device integrations or refining analytics algorithms.
        - Generates business intelligence reports to guide strategic decision-making for HealthFit's growth and development.

Logical Data Model

Entities and Attributes

1. **Users Table**
    - **Primary Key**: `UserID`
    - **Attributes**:
        - `FirstName`
        - `LastName`
        - `DateOfBirth`

        - `Gender`
        - `Email`
        - `PhoneNumber`
        - `SignupDate`
2. **Devices Table**
    - **Primary Key**: `DeviceID`
    - **Foreign Key**: `UserID`
    - **Attributes**:
        - `DeviceType` (e.g., fitness tracker, smartwatch)
        - `DeviceModel`
        - `RegistrationDate`
3. **HealthMetrics Table**
    - **Primary Key**: `MetricID`
    - **Foreign Keys**: `UserID`, `DeviceID`
    - **Attributes**:
        - `MetricType` (e.g., heart rate, sleep)
        - `MetricValue`
        - `Timestamp`
4. **ElectronicHealthRecords Table (EHR)**
    - **Primary Key**: `RecordID`
    - **Foreign Key**: `UserID`
    - **Attributes**:
        - `RecordType` (e.g., diagnosis, medication)
        - `RecordDetails`
        - `RecordDate`
5. **Recommendations Table**
    - **Primary Key**: `RecommendationID`
    - **Foreign Key**: `UserID`
    - **Attributes**:
        - `RecommendationDetails`
        - `GeneratedDate`
6. **ActivityLog Table**
    - **Primary Key**: `LogID`
    - **Foreign Key**: `UserID`
    - **Attributes**:
        - `ActivityType` (e.g., login, profile update)
        - `Timestamp`

Relationships Between Entities

1. **Users** ↔ **Devices**: One user can own multiple devices.
    - Relationship: One-to-Many
    - Foreign Key: `UserID` in `Devices`
2. **Users** ↔ **HealthMetrics**: One user can have numerous health metrics logged.

    - Relationship: One-to-Many
    - Foreign Key: `UserID` in `HealthMetrics`
3. **Devices** ↔ **HealthMetrics**: One device can generate various health metrics.
    - Relationship: One-to-Many
    - Foreign Key: `DeviceID` in `HealthMetrics`
4. **Users** ↔ **EHR**: One user can have multiple electronic health records.
    - Relationship: One-to-Many
    - Foreign Key: `UserID` in `EHR`
5. **Users** ↔ **Recommendations**: One user can have multiple personalized recommendations.
    - Relationship: One-to-Many
    - Foreign Key: `UserID` in `Recommendations`
6. **Users** ↔ **ActivityLog**: One user can have various activity logs.
    - Relationship: One-to-Many
    - Foreign Key: `UserID` in `ActivityLog`

Logical Data Model Diagram (Described in Text)

- **Users** are the central entity connected to:
    - **Devices** (via `UserID`)
    - **HealthMetrics** (via `UserID`)
    - **EHR** (via `UserID`)
    - **Recommendations** (via `UserID`)
    - **ActivityLog** (via `UserID`)
- **Devices** connect to **HealthMetrics** through `DeviceID`.

Database Objects

1. **Tables**
    - The primary storage mechanism for structured data within the database.
    - Tables include attributes (columns) to store data and primary/foreign keys to define relationships.
    - Key tables in the database:
        - **Users**
        - **Devices**
        - **HealthMetrics**
        - **ElectronicHealthRecords (EHR)**
        - **Recommendations**
        - **ActivityLog**
2. **Primary Keys**
    - Uniquely identify each record in a table.
    - Examples:
        - `UserID` in the **Users** table.
        - `DeviceID` in the **Devices** table.

        - `MetricID` in the **HealthMetrics** table.
3. **Foreign Keys**
    - Establish relationships between tables by referencing primary keys in other tables.
    - Examples:
        - `UserID` in **Devices**, **HealthMetrics**, **EHR**, **Recommendations**, and **ActivityLog**.
4. **Indexes**
    - Optimize query performance by enabling faster searches.
    - Key indexes:
        - Index on `UserID` in all related tables.
        - Index on `MetricType` and `Timestamp` in the **HealthMetrics** table for time-series analysis.
5. **Views**
    - Virtual tables that simplify complex queries by aggregating and summarizing data.
    - Examples:
        - A view combining user data and recent health metrics for real-time monitoring.
        - A view summarizing health recommendations and their outcomes.
6. **Stored Procedures**
    - Predefined SQL queries that automate recurring operations.
    - Examples:
        - A procedure to generate personalized health recommendations.
        - A procedure to archive old activity logs.
7. **Triggers**
    - Automated actions are executed based on events in the database.
    - Examples:
        - A trigger to log user activity when a new health metric is added.
        - A trigger to send notifications when certain health thresholds are crossed.
8. **Schemas**
    - Logical grouping of tables and other database objects for organization and access control.
    - Example:
        - Schema `HealthData` containing all the HealthTrack-related tables and objects.

Storage and File Attributes

1. **Storage Mechanisms**
    - Data will be stored using:
        - **Clustered Storage**: Ensures primary key data is stored in contiguous blocks for fast retrieval.
        - **Partitioning**: Health metrics and activity logs can be partitioned by date for better performance with time-based queries.
2. **File Attributes in the Database Solution**

- **Users Table**
  - Attributes: `UserID`, `FirstName`, `LastName`, `DateOfBirth`, `Gender`, `Email`, `PhoneNumber`, `SignupDate`
  - Storage: User profile data is stored as fixed-length or variable-length fields based on attribute type.
- **Devices Table**
  - Attributes: `DeviceID`, `UserID`, `DeviceType`, `DeviceModel`, `RegistrationDate`
  - Storage: Device registration data is stored with relationships to the **Users** table.
- **HealthMetrics Table**
  - Attributes: `MetricID`, `UserID`, `DeviceID`, `MetricType`, `MetricValue`, `Timestamp`
  - Storage: Time-series data with efficient indexing on `Timestamp`.
- **ElectronicHealthRecords Table**
  - Attributes: `RecordID`, `UserID`, `RecordType`, `RecordDetails`, `RecordDate`
  - Storage: Text or binary fields for medical records, depending on the format of `RecordDetails`.
- **Recommendations Table**
  - Attributes: `RecommendationID`, `UserID`, `RecommendationDetails`, `GeneratedDate`
  - Storage: Recommendations are stored as structured text.
- **ActivityLog Table**
  - Attributes: `LogID`, `UserID`, `ActivityType`, `Timestamp`
  - Storage: Log data stored in chronological order with an index on `Timestamp`.

Data Organization and Management

- **Normalization**: Ensures no redundant data and maintains integrity through relationships.
- **Backups and Recovery**: Regular backups will ensure data availability in case of system failures.
- **Retention Policies**:
  - Health metrics older than a specified time can be archived to reduce storage costs.
  - Activity logs can be retained for auditing purposes before being archived.

Scalability in the Proposed Database Design

The proposed database design for HealthTrack addresses scalability concerns by implementing strategies and features that ensure the database can handle increasing data volumes, user demands, and diverse data types as HealthFit Innovations grows.

Key Strategies for Scalability

1. **Database Partitioning**
    - **What It Is**: Dividing large tables into smaller, more manageable pieces (partitions) based on specific criteria, such as date or user region.
    - **Implementation in HealthTrack**:
        - **HealthMetrics Table**: Partitioned by `Timestamp` to manage time-series data efficiently.
        - **ActivityLog Table**: Partitioned by date ranges, allowing faster query performance for recent activity.
    - **Benefits**:
        - Improved query performance by reducing the search space for queries.
        - Simplifies archiving and deletion of old data without impacting the entire table.
2. **Indexing**
    - **What It Is**: Creating indexes on frequently queried columns to speed up data retrieval.
    - **Implementation in HealthTrack**:
        - Indexes on `UserID` in all related tables to quickly retrieve user-specific data.
        - Indexes on `Timestamp` and `MetricType` in the **HealthMetrics Table** for time-based and type-based queries.
    - **Benefits**:
        - Reduces the time required to fetch records from large tables.
        - Optimizes complex analytical queries for real-time health monitoring.
3. **Horizontal Scaling**
    - **What It Is**: Adding more servers to distribute data and workload.
    - **Implementation in HealthTrack**:
        - Use of sharding to distribute user data across multiple database nodes.
        - Each shard contains a subset of users, allowing for the parallel processing of queries.
    - **Benefits**:
        - Supports a growing user base without overwhelming a single server.
        - Increases the overall processing capacity of the database.
4. **Database Normalization**
    - **What It Is**: Structuring data to minimize redundancy and dependencies.
    - **Implementation in HealthTrack**:
        - Tables like **Users**, **Devices**, and **HealthMetrics** are designed with proper relationships to avoid duplication.
    - **Benefits**:
        - Reduces storage requirements.
        - Simplifies updates and ensures data consistency.
5. **Read-Write Separation**
    - **What It Is**: Using separate database instances for read and write operations.
    - **Implementation in HealthTrack**:

        - A master database handles write operations (e.g., adding new health metrics), while replica databases handle read operations (e.g., generating reports or recommendations).
    - **Benefits**:
        - Prevents bottlenecks caused by simultaneous read and write operations.
        - Enhances system performance for both data ingestion and analytics.
6. **Cloud-Based Scaling**
    - **What It Is**: Leveraging cloud database solutions for dynamic scaling based on demand.
    - **Implementation in HealthTrack**:
        - Hosting the database on a cloud platform with auto-scaling capabilities to handle spikes in data ingestion or user activity.
    - **Benefits**:
        - On-demand scalability without upfront hardware investments.
        - Ensures high availability and disaster recovery.
7. **Archiving and Retention Policies**
    - **What It Is**: Moving older or less frequently accessed data to a separate storage system.
    - **Implementation in HealthTrack**:
        - Archiving old health metrics and activity logs to a data warehouse or cheaper storage tiers.
    - **Benefits**:
        - Keeps the primary database lean and optimized for active data.
        - Reduces storage and operational costs.

Alignment with the Scenario

1. **Handling Diverse and Dynamic Data**
    - The design accommodates wearable device data, electronic health records, and user-reported outcomes.
    - The database efficiently integrates and retrieves diverse data types by leveraging a normalized schema and indexing.
2. **Real-Time Insights and Predictive Analytics**
    - Indexing, partitioning, and read-write separation enable real-time performance, which is crucial for HealthTrack's predictive analytics and monitoring features.
3. **Support for Growing User Base**
    - Horizontal scaling, cloud deployment, and sharding ensure that the database can grow with the user base without sacrificing performance.
4. **Future-Ready Design**
    - Using a relational database supports integration with advanced tools like machine learning models or additional data sources, ensuring scalability for future innovations.

1. Authentication and Authorization

- **Role-Based Access Control (RBAC)**:

    - Assign specific roles (e.g., admin, healthcare provider, user) with clearly defined access permissions.
    - Example: Only administrators can update the database schema, while healthcare providers can view patient records.
- **Multi-Factor Authentication (MFA)**:
    - To prevent unauthorized access, users should be required to verify their identity through an additional factor, such as a code sent to their phone.
- **Secure Password Storage**:
    - Store passwords using strong hashing algorithms, such as bcrypt or Argon2, to prevent the exposure of plaintext passwords in case of a breach.

2. Data Encryption

- **Encryption at Rest**:
    - Encrypt all data stored in the database using industry-standard encryption algorithms like AES-256.
    - Applies to tables containing sensitive data, such as user profiles, health metrics, and medical records.
- **Encryption in Transit**:
    - Use TLS (Transport Layer Security) to encrypt data transmitted between the database and clients (e.g., web applications, wearables).
- **Field-Level Encryption**:
    - Encrypt particularly sensitive fields, such as `RecordDetails,` in the **EHR** table, to add an extra layer of protection.

3. Data Anonymization and Masking

- **Anonymization**:
    - Personal identifiers (e.g., names and emails) are removed or transformed when data is used for analysis or shared with third parties.
    - Use techniques like hashing or pseudonymization for non-identifiable datasets.
- **Data Masking**:
    - Mask sensitive data (e.g., showing only the last four digits of a phone number) for non-administrative users or during testing and development.

4. Audit Logging and Monitoring

- **Audit Logs**:
    - Maintain detailed logs of all database activities, such as data access, modifications, and user logins.
    - Logs should include information like the `UserID`, timestamp, and operation performed.
- **Anomaly Detection**:
    - Automate monitoring tools to identify unusual patterns, such as multiple failed login attempts or unauthorized data access.

5. Data Backup and Recovery

- **Regular Backups**:
  - Perform regular, automated backups of the database to a secure location.
  - Use encryption for all backup files.
- **Disaster Recovery Plan**:
  - Implement a robust disaster recovery plan to restore data quickly in case of hardware failure, cyberattacks, or other incidents.

6. Compliance with Healthcare Regulations

- **HIPAA Compliance**:
  - Ensure the database meets the requirements of the Health Insurance Portability and Accountability Act (HIPAA), including access control, encryption, and logging.
- **GDPR Compliance** (if applicable):
  - Implement measures to meet the General Data Protection Regulation (GDPR), such as user consent for data collection and the right to request data deletion.

7. Security Best Practices

- **Least Privilege Principle**:
  - Ensure users and applications have access only to the data they need for their role.
  - Example: A healthcare provider can view health metrics but cannot access database configurations.
- **Database Firewall**:
  - Use firewalls to prevent unauthorized access to the database from external networks.
- **Regular Updates and Patching**:
  - Apply security patches and updates to the database management system (DBMS) promptly to address vulnerabilities.
- **Intrusion Detection and Prevention Systems (IDPS)**:
  - Use IDPS tools to detect and prevent malicious activities targeting the database.

8. Secure Development Practices

- **Secure Query Writing**:
  - Use parameterized queries or prepared statements to prevent SQL injection attacks.
  - Example: Avoid dynamic SQL queries that concatenate user input directly.
- **Development and Testing Environment Isolation**:
  - Keep development and testing environments separate from production to avoid accidental data exposure.

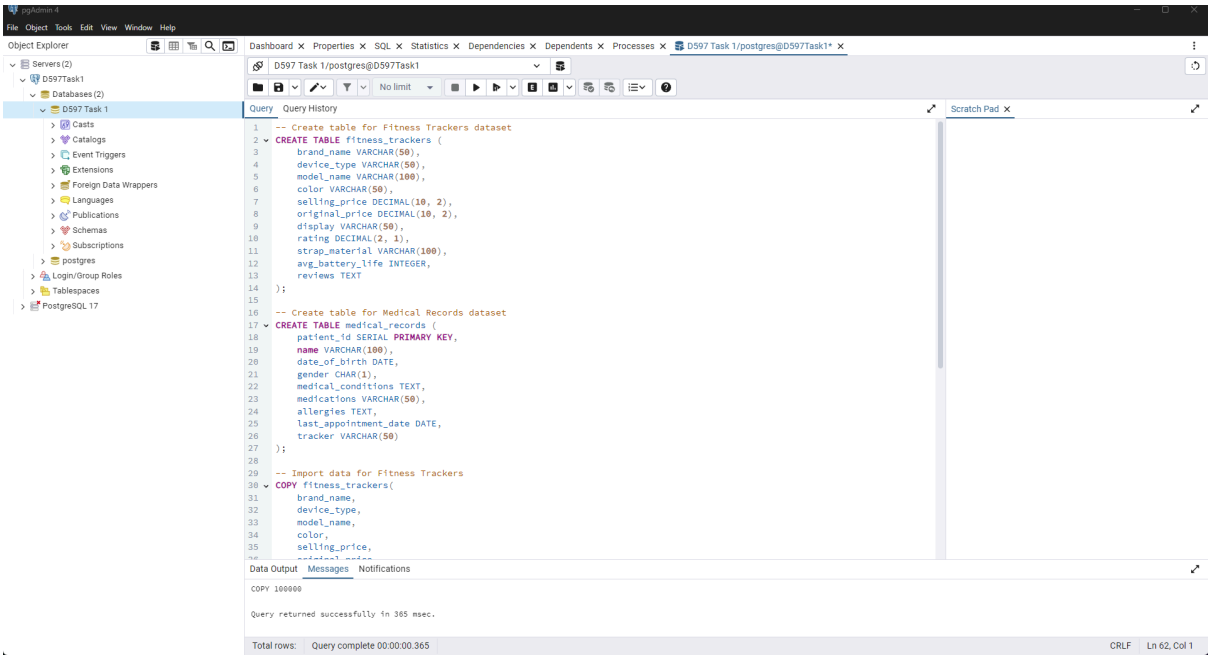
9. Data Retention Policies

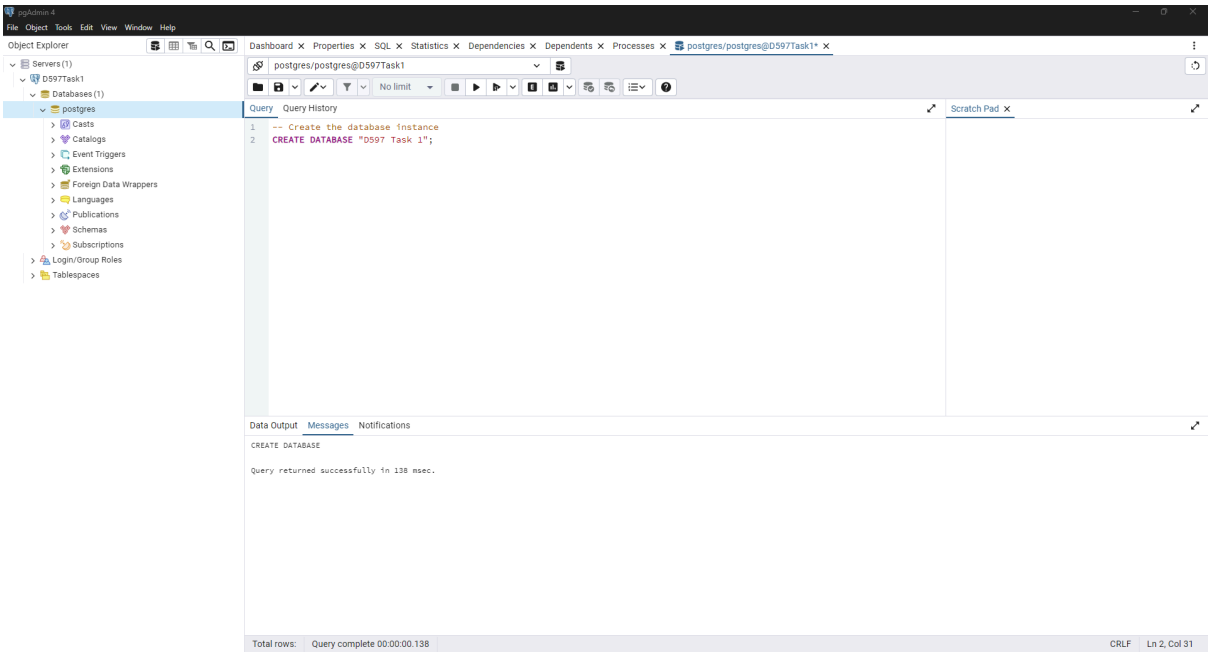
- **Retention Limits**:
    - Define policies for how long user data is stored and archive or delete old data no longer needed.
    - Example: Automatically archive health metrics older than one year.
- **Secure Deletion**:
    - Use secure deletion methods to ensure that deleted data cannot be recovered.

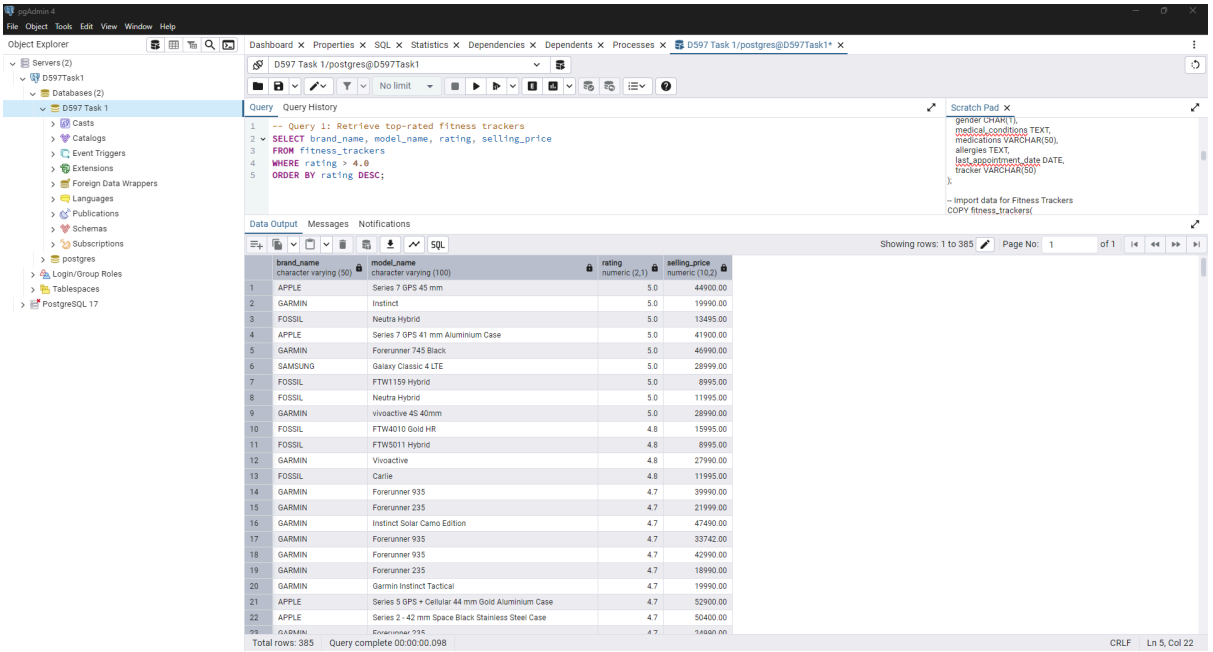
10. User Transparency and Control

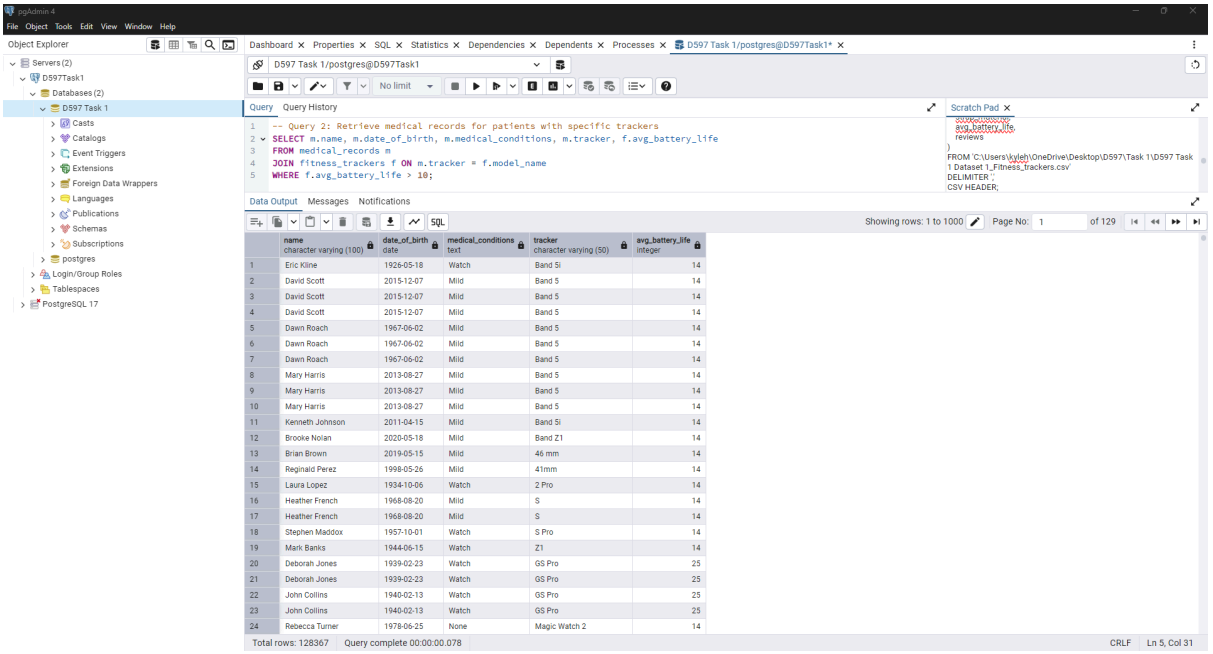
- **Privacy Policies**:
    - Communicate how user data is collected, stored, and used.
    - Allow users to access their data, update their information, or revoke consent for data collection.
- **Data Access Logs**:
    - Provide users with access logs showing who has viewed or modified their data.

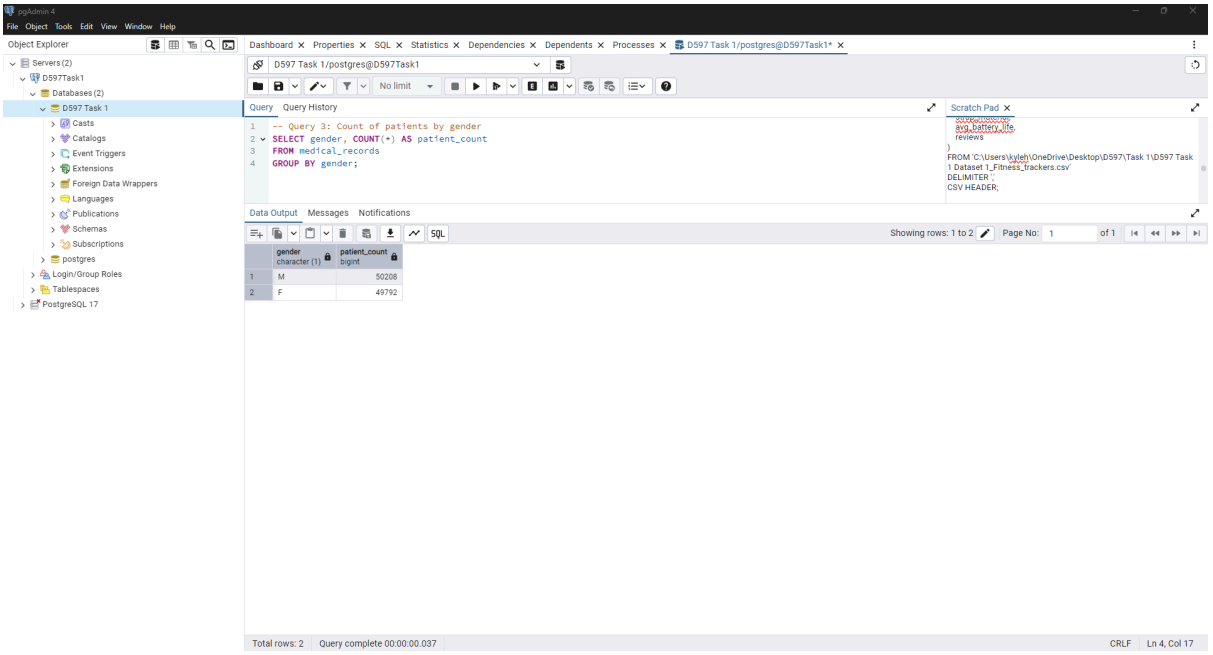
Part 2: Implementation

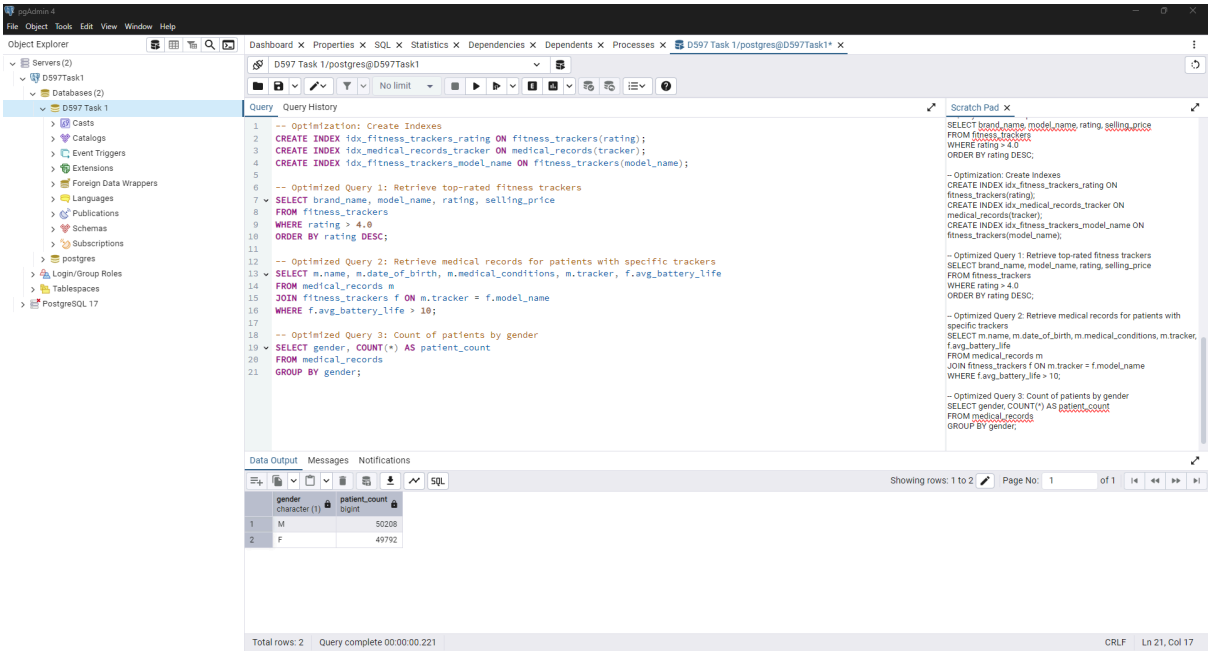


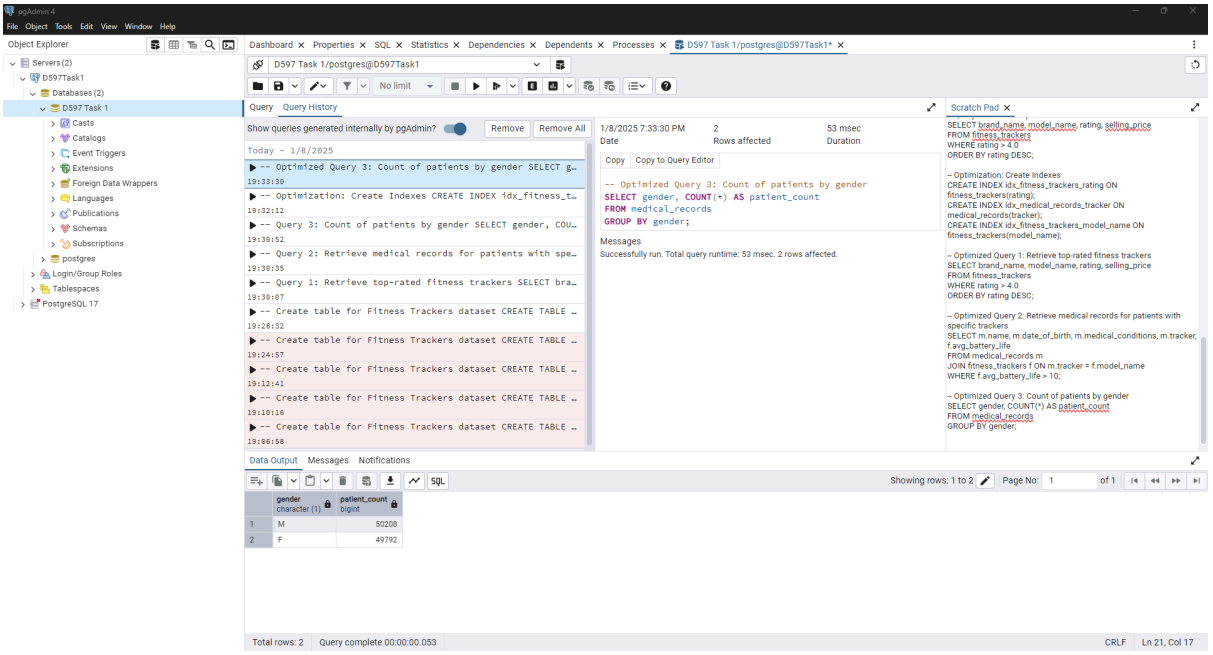



pgAdmin 4
File Object Tools Edit View Window Help
Object Explorer
Servers (2)
D597Task1
Databases (2)
D597 Task 1
Casts
Catalogs
Event Triggers
Extensions
Foreign Data Wrappers
Languages
Publications
Schemas
Subscriptions
postgres
Login/Group Roles
Tablespaces
PostgreSQL 17
Dashboard
Properties
SQL
Statistics
Dependencies
Dependents
Processes
D597 Task 1/postgres@D597Task1*
D597 Task 1/postgres@D597Task1
No limit
Query
Query History
-- Query 1: Retrieve top-rated fitness trackers
SELECT brand_name, model_name, rating, selling_price
FROM fitness_trackers
WHERE rating > 4.0
ORDER BY rating DESC;
Scratch Pad
gender CHAR(1),
medical_conditions TEXT,
medications VARCHAR(50),
allergies TEXT,
last_appointment_date DATE,
tracker VARCHAR(50)
);
-- Import data for Fitness Trackers
COPY fitness_trackers(
Data Output
Messages
Notifications
SQL
Showing rows: 1 to 385
Page No: 1
of 1
brand_name character varying (50)
model_name character varying (100)
rating numeric (2,1)
selling_price numeric (10,2)
1 APPLE Series 7 GPS 45 mm 5.0 44900.00
2 GARMIN Instinct 5.0 19990.00
3 FOSSIL Neutra Hybrid 5.0 13495.00
4 APPLE Series 7 GPS 41 mm Aluminium Case 5.0 41900.00
5 GARMIN Forerunner 745 Black 5.0 46990.00
6 SAMSUNG Galaxy Classic 4 LTE 5.0 28999.00
7 FOSSIL FTW1159 Hybrid 5.0 8995.00
8 FOSSIL Neutra Hybrid 5.0 11995.00
9 GARMIN vivoactive 4S 40mm 5.0 28990.00
10 FOSSIL FTW4010 Gold HR 4.8 15995.00
11 FOSSIL FTW5011 Hybrid 4.8 8995.00
12 GARMIN Vivoactive 4.8 27990.00
13 FOSSIL Carlie 4.8 11995.00
14 GARMIN Forerunner 935 4.7 39990.00
15 GARMIN Forerunner 235 4.7 21999.00
16 GARMIN Instinct Solar Camo Edition 4.7 47490.00
17 GARMIN Forerunner 935 4.7 33742.00
18 GARMIN Forerunner 935 4.7 42990.00
19 GARMIN Forerunner 235 4.7 18990.00
20 GARMIN Garmin Instinct Tactical 4.7 19990.00
21 APPLE Series 5 GPS + Cellular 44 mm Gold Aluminium Case 4.7 52900.00
22 APPLE Series 2 - 42 mm Space Black Stainless Steel Case 4.7 50400.00
Total rows: 385
Query complete 00:00:00.098
CRLF
Ln 5, Col 22


pgAdmin 4
File Object Tools Edit View Window Help
Object Explorer
Servers (2)
D597Task1
Databases (2)
D597 Task 1
Casts
Catalogs
Event Triggers
Extensions
Foreign Data Wrappers
Languages
Publications
Schemas
Subscriptions
postgres
Login/Group Roles
Tablespaces
PostgreSQL 17
Dashboard
Properties
SQL
Statistics
Dependencies
Dependents
Processes
D597 Task 1/postgres@D597Task1*
D597 Task 1/postgres@D597Task1
No limit
Query
Query History
-- Query 2: Retrieve medical records for patients with specific trackers
SELECT m.name, m.date_of_birth, m.medical_conditions, m.tracker, f.avg_battery_life
FROM medical_records m
JOIN fitness_trackers f ON m.tracker = f.model_name
WHERE f.avg_battery_life > 10;
Scratch Pad
avg_battery_life,
reviews
)
FROM 'C:\Users\kyleh\OneDrive\Desktop\D597\Task 1\D597 Task 1 Dataset 1_Fitness_trackers.csv'
DELIMITER ','
CSV HEADER;
Data Output
Messages
Notifications
SQL
Showing rows: 1 to 1000
Page No: 1
of 129
name character varying (100)
date_of_birth date
medical_conditions text
tracker character varying (50)
avg_battery_life integer
1 Eric Kline 1926-05-18 Watch Band Si 14
2 David Scott 2015-12-07 Mild Band 5 14
3 David Scott 2015-12-07 Mild Band 5 14
4 David Scott 2015-12-07 Mild Band 5 14
5 Dawn Roach 1967-06-02 Mild Band 5 14
6 Dawn Roach 1967-06-02 Mild Band 5 14
7 Dawn Roach 1967-06-02 Mild Band 5 14
8 Mary Harris 2013-08-27 Mild Band 5 14
9 Mary Harris 2013-08-27 Mild Band 5 14
10 Mary Harris 2013-08-27 Mild Band 5 14
11 Kenneth Johnson 2011-04-15 Mild Band Si 14
12 Brooke Nolan 2020-05-18 Mild Band Z1 14
13 Brian Brown 2019-05-15 Mild 46 mm 14
14 Reginald Perez 1998-05-26 Mild 41mm 14
15 Laura Lopez 1934-10-06 Watch 2 Pro 14
16 Heather French 1968-08-20 Mild S 14
17 Heather French 1968-08-20 Mild S 14
18 Stephen Maddox 1957-10-01 Watch S Pro 14
19 Mark Banks 1944-06-15 Watch Z1 14
20 Deborah Jones 1939-02-23 Watch GS Pro 25
21 Deborah Jones 1939-02-23 Watch GS Pro 25
22 John Collins 1940-02-13 Watch GS Pro 25
23 John Collins 1940-02-13 Watch GS Pro 25
24 Rebecca Turner 1978-06-25 None Magic Watch 2 14
Total rows: 128367
Query complete 00:00:00.078
CRLF
Ln 5, Col 31


pgAdmin 4
File Object Tools Edit View Window Help
Object Explorer
Servers (2)
D597Task1
Databases (2)
D597 Task 1
Casts
Catalogs
Event Triggers
Extensions
Foreign Data Wrappers
Languages
Publications
Schemas
Subscriptions
postgres
Login/Group Roles
Tablespaces
PostgreSQL 17
Dashboard
Properties
SQL
Statistics
Dependencies
Dependents
Processes
D597 Task 1/postgres@D597Task1*
D597 Task 1/postgres@D597Task1
No limit
Query
Query History
-- Query 3: Count of patients by gender
SELECT gender, COUNT(*) AS patient_count
FROM medical_records
GROUP BY gender;
Scratch Pad
avg_battery_life,
reviews
)
FROM 'C:\Users\kyleh\OneDrive\Desktop\D597\Task 1\D597 Task 1 Dataset 1_Fitness_trackers.csv'
DELIMITER ','
CSV HEADER;
Data Output
Messages
Notifications
SQL
Showing rows: 1 to 2
Page No: 1
of 1
gender character (1)
patient_count bigint
1 M 50208
2 F 49792
Total rows: 2
Query complete 00:00:00.037
CRLF
Ln 4, Col 17


pgAdmin 4
File Object Tools Edit View Window Help
Object Explorer
Servers (2)
D597Task1
Databases (2)
D597 Task 1
Casts
Catalogs
Event Triggers
Extensions
Foreign Data Wrappers
Languages
Publications
Schemas
Subscriptions
postgres
Login/Group Roles
Tablespaces
PostgreSQL 17
Dashboard
Properties
SQL
Statistics
Dependencies
Dependents
Processes
D597 Task 1/postgres@D597Task1*
D597 Task 1/postgres@D597Task1
No limit
Query
Query History
-- Optimization: Create Indexes
CREATE INDEX idx_fitness_trackers_rating ON fitness_trackers(rating);
CREATE INDEX idx_medical_records_tracker ON medical_records(tracker);
CREATE INDEX idx_fitness_trackers_model_name ON fitness_trackers(model_name);
-- Optimized Query 1: Retrieve top-rated fitness trackers
SELECT brand_name, model_name, rating, selling_price
FROM fitness_trackers
WHERE rating > 4.0
ORDER BY rating DESC;
-- Optimized Query 2: Retrieve medical records for patients with specific trackers
SELECT m.name, m.date_of_birth, m.medical_conditions, m.tracker, f.avg_battery_life
FROM medical_records m
JOIN fitness_trackers f ON m.tracker = f.model_name
WHERE f.avg_battery_life > 10;
-- Optimized Query 3: Count of patients by gender
SELECT gender, COUNT(*) AS patient_count
FROM medical_records
GROUP BY gender;
Scratch Pad
SELECT brand_name, model_name, rating, selling_price
FROM fitness_trackers
WHERE rating > 4.0
ORDER BY rating DESC;
-- Optimization: Create Indexes
CREATE INDEX idx_fitness_trackers_rating ON fitness_trackers(rating);
CREATE INDEX idx_medical_records_tracker ON medical_records(tracker);
CREATE INDEX idx_fitness_trackers_model_name ON fitness_trackers(model_name);
-- Optimized Query 1: Retrieve top-rated fitness trackers
SELECT brand_name, model_name, rating, selling_price
FROM fitness_trackers
WHERE rating > 4.0
ORDER BY rating DESC;
-- Optimized Query 2: Retrieve medical records for patients with specific trackers
SELECT m.name, m.date_of_birth, m.medical_conditions, m.tracker, f.avg_battery_life
FROM medical_records m
JOIN fitness_trackers f ON m.tracker = f.model_name
WHERE f.avg_battery_life > 10;
-- Optimized Query 3: Count of patients by gender
SELECT gender, COUNT(*) AS patient_count
FROM medical_records
GROUP BY gender;
Data Output
Messages
Notifications
SQL
Showing rows: 1 to 2
Page No: 1
of 1
gender character (1)
patient_count bigint
1 M 50208
2 F 49792
Total rows: 2
Query complete 00:00:00.221
CRLF
Ln 21, Col 17



1/8/2025 7:33:30 PM | 2 | 53 msec

Date | Rows affected | Duration

Copy | Copy to Query Editor

```
-- Optimized Query 3: Count of patients by gender
SELECT gender, COUNT(*) AS patient_count
FROM medical_records
GROUP BY gender;
```

Messages

Successfully run. Total query runtime: 53 msec. 2 rows affected.

1/8/2025 7:32:12 PM	2	220 msec
Date	Rows affected	Duration

Copy | Copy to Query Editor

```
-- Optimization: Create Indexes
CREATE INDEX idx_fitness_trackers_rating ON fitness_trackers(ra
CREATE INDEX idx_medical_records_tracker ON medical_records(tra
CREATE INDEX idx_fitness_trackers_model_name ON fitness_tracker

-- Optimized Query 1: Retrieve top-rated fitness trackers
SELECT brand_name, model_name, rating, selling_price
FROM fitness_trackers
WHERE rating > 4.0
ORDER BY rating DESC;

-- Optimized Query 2: Retrieve medical records for patients wi
SELECT m.name, m.date_of_birth, m.medical_conditions, m.tracke
FROM medical_records m
JOIN fitness_trackers f ON m.tracker = f.model_name
WHERE f.avg_battery_life > 10;

-- Optimized Query 3: Count of patients by gender
SELECT gender, COUNT(*) AS patient_count
FROM medical_records
GROUP BY gender;
```

Messages

Successfully run. Total query runtime: 220 msec. 2 rows affected.

Course |. (n.d.). https://apps.cgp-oex.wgu.edu/wgulearning/course/course-v1:WGUx+OEX0343+v01/block-v1:WGUx+OEX0343+v01+type@sequential+block@543516264ee84ad4b89a1449a5d2db90/block-v1:WGUx+OEX0343+v01+type@vertical+block@63e2765421dc476a8611a55973cafd50

Lee, C. (2025, January 2). What is a logical data model? *GoodData*. https://www.gooddata.com/blog/how-build-logical-data-models-scale-analytical-applications/

Yen, L. (2024, May 24). *Logical vs Physical Data Model: A Comprehensive Guide*. Datamation. https://www.datamation.com/big-data/logical-vs-physical-data-model/

Reiber, L. (2024, May 7). *Guided Project: Answering Business Questions using SQL – Dataquest*. Dataquest. https://www.dataquest.io/projects/guided-project-a-answering-business-questions-using-sql/

How to use DML statements to manipulate data in SQL databases. (2024, December 9). https://chat2db.ai/resources/blog/how-to-use-dml-statements-to-manipulate-data-in-sql-databases