

2507023_Shahriar_khan_Assignment1_Database_System

by Shahriar Khan

Submission date: 08-Jan-2026 02:55AM (UTC+0000)

Submission ID: 272477448

File name:

67167_Shahriar_Khan_2507023_Shahriar_khan_Assignment1_Database_System_709999_1265481113.docx
(1.74M)

Word count: 2513

Character count: 16882



Prifysgol Cymru
Y Drindod Dewi Sant
University of Wales
Trinity Saint David

DragonWave Renewables Ltd (DWR) Database Design & Implementation

Module: Database Systems

Module Code: ACCA7002

Submitted To:

Divya Chidambaram
Lecturer, UWTSB

Submitted By:

Name: Shahriar Khan
Student ID: 2507023

Submission Date: 09 January 2026

1	Introduction
2	Significance and business context
3	Database design  1
4	Entity analysis
5	Relationship analysis
6	Normalization process
7	Database decisions
8	Database schema creation
9	Sample data insertion
10	Trigger implementation
11	Stored procedure development
12	Stored procedure and transaction management
13	Performance tuning and enhancement
14	Conclusion

Introduction:

Case scenario: DWR Solar Solutions is a growing renewable energy company specializing in residential and commercial solar panel installations. The company manages a complex ecosystem involving customers, solar system products, certified installers, installation orders, maintenance schedules, and customer feedback. The assignment requires a database for the efficient management of the company.

They manage:

- Customer who buys solar system
- Different Solar system models with various qualifications
- Certified installer who does the installation work
- Installation order that tracks each job
- Maintenance record for system upkeep
- Customer review for feedback



Significance and business context

The database system implementation is necessary. If the company uses spreadsheets or paper for management, it arises several problems like data errors, loss of information, slow processes etc. After implementation of database, it gives benefits like data consistency, easy tracking, quick information access, quality control, better customer service, inventory management. It also helps the company make decisions to improve their service.



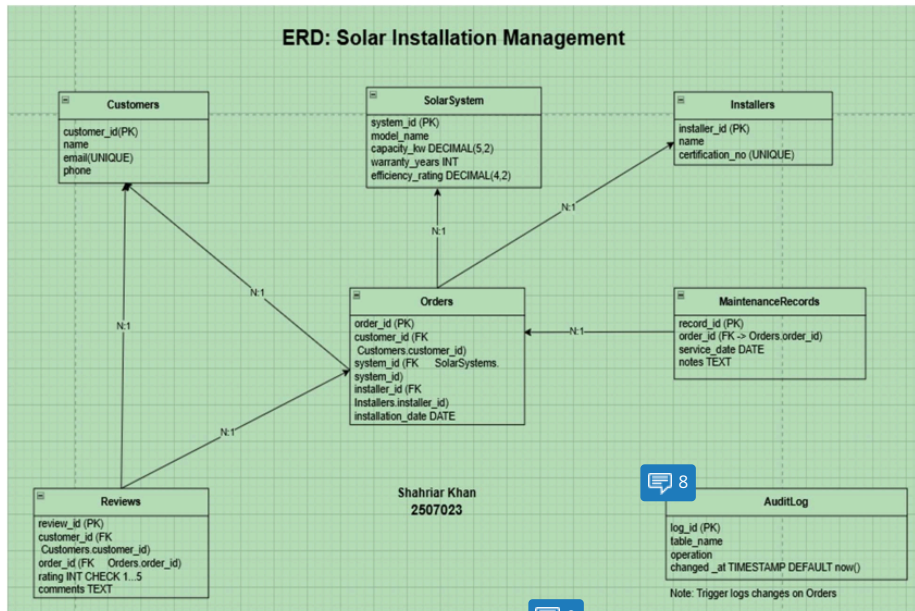
Project Objective

The primary objectives of database implementation project are:

1. Design a normalized scheme that eliminates data redundancy
2. Implement constraint to provide data validity
3. Develop transactional procedures to maintain data consistency during complex operation
4. Optimize query performance by indexing and schema design
5. Provide Reliable backup and recovery capabilities through transaction management
6. Enable scalability for future business growth
7. Create audit mechanism for tracking data changes



Database Design: ER diagram



This is a blueprint for DRW solutions that shows how all their business information connects. How they connect:

- Customer → Orders (One-to-Many)**
One customer can have many installations order
- Solar Systems → Orders (One-to-Many)**
One solar model can be installed in many orders
- Orders → Maintenance Records (One-to-Many)**
One order can have many maintenances visit
- Customer → Reviews (One-to-Many)**
One customer can leave many reviews
- Installers → Orders (One-to-Many)**
One installer can work on many orders.

Audit System

Audit Log - Special tracking system

- Automatically records when orders are created, changed, or deleted
- Like a security camera for database changes
- Shows about who changed what and when

How It Works in Practice

When a customer buys a solar system:

1. Their information goes on the Customers table
2. The solar model goes in Solar Systems
3. The installer assignment goes in Installers
4. All this connects to the Orders table
5. Later maintenance goes in Maintenance Records
6. Customer feedback goes in Reviews
7. All changes are tracked in Audit Log

Why This Design Works

- Efficient - No repeated information
- Accurate - Rules prevent wrong data entry
- Scalable - Can handle thousands of customers
- Connected - Everything links together logically
- Trackable - Can follow the complete history

This ERD is like a well-organized filing system where everything has its place, and you can easily find connections between different pieces of information.

Entity Analysis:

1. Customers Entity

Primary key: customer_id (SERIAL, auto-incrementing)

Attributes:

- name: Customer full name (VARCHAR (100))
- email: Unique contact email with UNIQUE constraint
- phone: Contact phone number (VARCHAR (20))

2. Solar Systems Entity

Primary Key: system_id (SERIAL)

Attributes:

- model_name: Product model identifier (VARCHAR (100))
- Capacity_kw: System capacity in kilowatts (DECIMAL (5,2))
- warranty_years: Manufacturer warranty duration (INT)
- efficiency_rating: Performance efficiency percentage (DECIMAL (4,2))

3. Installers Entity

Primary Key: installer_id (SERIAL)

Attributes:

- name: Installer's full name (VARCHAR (100))
- certification_no: Unique certification number with UNIQUE constraint

4. Orders Entity

Primary Key: order_id (SERIAL)

Foreign Keys:

- customer_id references Customers(customer_id)
- system_id references SolarSystems(system_id)
- installer_id references Installers(installer_id)

Attributes:

- installation_date: Scheduled installation date (DATE)

5. MaintenanceRecords Entity

Primary Key:

- record_id (SERIAL)

Foreign Key:

- order_id references Orders(order_id)

Attributes:

- service_date: Maintenance service date (DATE)
- notes: Service details and observations (TEXT)

6. Reviews Entity

Primary Key:

- review_id (SERIAL)

Foreign Keys:

- customer_id references Customers(customer_id)
- order_id references Orders(order_id)

Attributes:

- rating: Customer rating (1-5 with CHECK constraint)
- comments: Customer feedback (TEXT)

Relationship Analysis

Customer-Order Relationship (One-to-Many)

Connection: Each customer can place multiple installation orders, but each order belongs to only one customer.

How it works:

- The customer_id from the **Customers** table is stored in each **Orders** record

Why this works for DWR:

- Easy to see all orders from one customer
- Customers can add more solar systems later
- Simple billing and warranty tracking per customer

SolarSystem-Order Relationship (One-to-Many)

Connection: One solar system model can be installed in many orders, but each order is only one system model.

How it works:

- The system_id from **SolarSystems** is stored in each **Orders** record

Why this works for DWR:

- Track which solar models are most popular
- Manage inventory of different system types
- Standardized warranty tracking by model

Installer-Order Relationship (One-to-Many)

Connection: One installer can complete many orders, but each order is assigned to only one installer.

How it works:

- The `installer_id` from **Installers** is stored in each **Orders** record

Why this works for DWR:

- Balance workload among different installers
- Track installer performance and quality
- Schedule maintenance with the right technician

Order-Maintenance Relationship (One-to-Many)

Connection: Each order can have multiple maintenance records, but each maintenance record belongs to only one order.

How it works:

- The `order_id` from **Orders** is stored in each **MaintenanceRecords** entry

Why this works for DWR:

- Complete service history for each installation
- Schedule regular maintenance automatically
- Track warranty claims and service needs

Customer-Review Relationship (One-to-Many)

Connection: Each customer can submit multiple reviews, but each review comes from only one customer.

How it works:

- The `customer_id` from **Customers** is stored in each **Reviews** entry

Why this works for DWR:

- Collect feedback from customers over time
- Identify satisfied customers for referrals
- Track customer satisfaction trends

Order-Review Relationship (One-to-One through Customer)

Connection: Each review is linked to both a customer AND a specific order.

How it works:

- Reviews store BOTH customer_id AND order_id
- This connects feedback to the exact installation that was reviewed

Why this works for DWR:

- Know exactly which installation is being reviewed
- Connect feedback to specific installers and systems
- Improve specific services based on feedback

This relationship design ensures DWR can:

- Track everything from customer to installation to maintenance
- Connect all parts of their business
- Generate accurate reports and histories
- Provide excellent customer service by having complete information

Normalization Process

The database design follows Third Normal Form (3NF) principles:

First Normal Form (1NF):

All tables have primary keys

- Atomic values in each column
- No repeating groups
- Consistent data types

Second Normal Form (2NF):

- All non-key attributes fully dependent on primary keys
- Partial dependencies eliminated

Third Normal Form (3NF):

- No transitive dependencies
- All non-key attributes mutually independent



Design decisions:

1. SERIAL Primary Keys

- Used SERIAL data type for all primary keys

Justification:

- Automatic unique value generation
- PostgreSQL optimization for sequential access
- Simplified application logic
- Industry standard practice for surrogate keys

2. VARCHAR Length Considerations

- VARCHAR(100) for names, VARCHAR(20) for phone numbers

Justification:

- Adequate for international name formats
- Phone numbers accommodate country codes and extensions
- Balance between storage efficiency and flexibility

3. DECIMAL Precision for Technical Data

- DECIMAL(5,2) for capacity, DECIMAL(4,2) for efficiency

Justification:

- Sufficient precision for solar industry requirements
- Accurate warranty and performance calculations
- Standard industry measurement units

4. DATE Data Type for Temporal Data

- DATE type for installation and service dates

Justification:

- Temporal query optimization
- Built-in date functions availability
- Clear semantic meaning

5. TEXT Data Type for Notes and Comments

- TEXT type for free-form fields

Justification:

- Unlimited length for detailed descriptions
- PostgreSQL optimization for large text fields
- Appropriate for unstructured data

6. Check Constraint for Ratings
 - CHECK (rating BETWEEN 1 AND 5)

Justification:

- Data validation at database level
- Application-independent business rule enforcement
- Improved data quality

This normalized design ensures data integrity, minimizes redundancy, and provides a scalable foundation for DWR's growing business needs while maintaining optimal performance through strategic design decisions.

Database Schema Creation

The database schema was implemented using PostgreSQL 13+, chosen for its robustness, ACID compliance, and advanced features suitable for transactional systems. The implementation follows a systematic approach to table creation, ensuring proper dependencies and relationship establishment.

Key Implementation Decisions:

SERIAL Primary Keys: Each table uses SERIAL for automatic primary key generation, ensuring uniqueness and simplifying application logic.

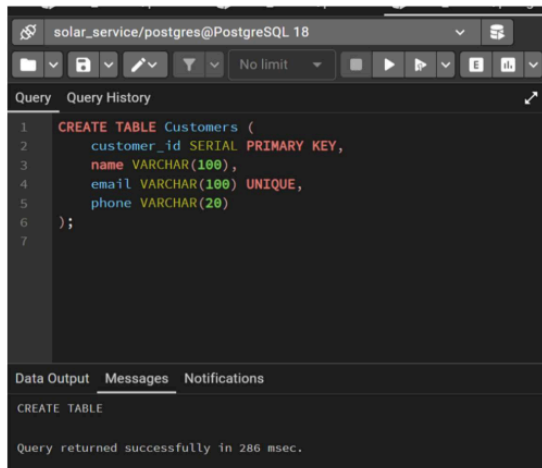
Appropriate Data Types:

- VARCHAR for variable-length text with reasonable limits
- DECIMAL for precise numeric calculations (critical for solar capacity)
- INT for whole numbers (warranty years, ratings)
- DATE for temporal data with built-in date functions
- TEXT for unlimited-length descriptive fields

Sequential Creation Order:

- Independent tables first (Customers, SolarSystems, Installers)
- Dependent tables next (Orders)
- Finally, secondary dependent tables (MaintenanceRecords, Reviews)

Customer table creation:



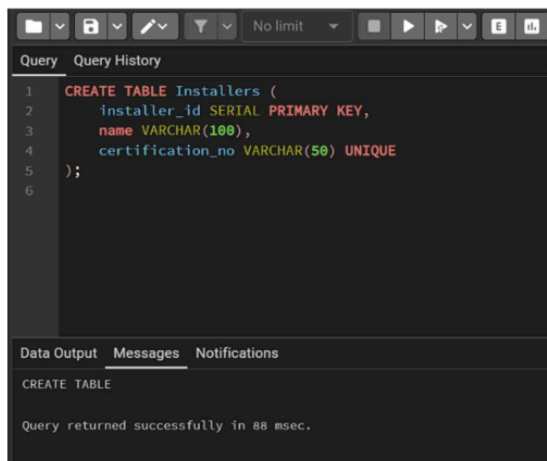
The screenshot shows a PostgreSQL query editor window titled 'solar_service/postgres@PostgreSQL 18'. The 'Query' tab is active, displaying a SQL statement to create a table named 'Customers'. The table has four columns: 'customer_id' (SERIAL PRIMARY KEY), 'name' (VARCHAR(100)), 'email' (VARCHAR(100) UNIQUE), and 'phone' (VARCHAR(20)). The 'Messages' tab is selected at the bottom, showing the command 'CREATE TABLE' and a confirmation message: 'Query returned successfully in 286 msec.'

```
1 CREATE TABLE Customers (  
2     customer_id SERIAL PRIMARY KEY,  
3     name VARCHAR(100),  
4     email VARCHAR(100) UNIQUE,  
5     phone VARCHAR(20)  
6 );  
7
```

CREATE TABLE

Query returned successfully in 286 msec.

Installer table creation:



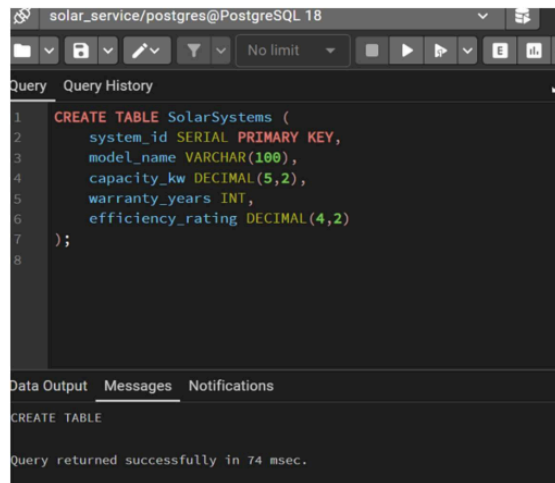
The screenshot shows a PostgreSQL query editor window titled 'solar_service/postgres@PostgreSQL 18'. The 'Query' tab is active, displaying a SQL statement to create a table named 'Installers'. The table has three columns: 'installer_id' (SERIAL PRIMARY KEY), 'name' (VARCHAR(100)), and 'certification_no' (VARCHAR(50) UNIQUE). The 'Messages' tab is selected at the bottom, showing the command 'CREATE TABLE' and a confirmation message: 'Query returned successfully in 88 msec.'

```
1 CREATE TABLE Installers (  
2     installer_id SERIAL PRIMARY KEY,  
3     name VARCHAR(100),  
4     certification_no VARCHAR(50) UNIQUE  
5 );  
6
```

CREATE TABLE

Query returned successfully in 88 msec.

Solar system table creation:



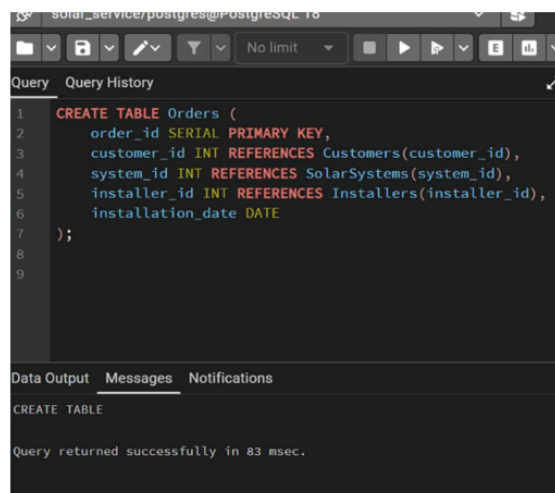
The screenshot shows a PostgreSQL query editor window titled 'solar_service/postgres@PostgreSQL 18'. The query editor has tabs for 'Query' and 'Query History'. The 'Query' tab is active, displaying a SQL command to create a table named 'SolarSystems'. The command is as follows:

```
1 CREATE TABLE SolarSystems (  
2     system_id SERIAL PRIMARY KEY,  
3     model_name VARCHAR(100),  
4     capacity_kw DECIMAL(5,2),  
5     warranty_years INT,  
6     efficiency_rating DECIMAL(4,2)  
7 );  
8
```

Below the query editor, there are tabs for 'Data Output', 'Messages', and 'Notifications'. The 'Messages' tab is active, showing the output of the query:

```
CREATE TABLE  
  
Query returned successfully in 74 msec.
```

Order table creation:



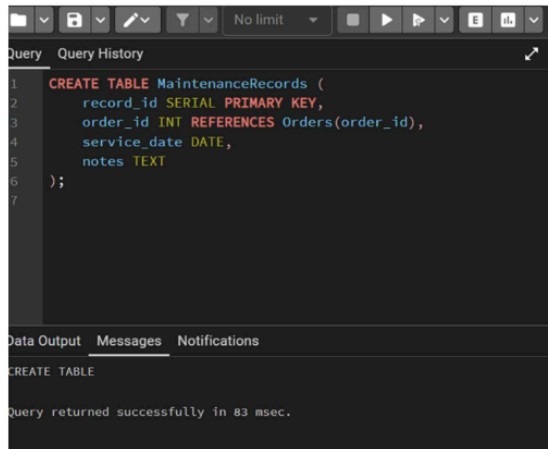
The screenshot shows a PostgreSQL query editor window titled 'solar_service/postgres@PostgreSQL 18'. The query editor has tabs for 'Query' and 'Query History'. The 'Query' tab is active, displaying a SQL command to create a table named 'Orders'. The command is as follows:

```
1 CREATE TABLE Orders (  
2     order_id SERIAL PRIMARY KEY,  
3     customer_id INT REFERENCES Customers(customer_id),  
4     system_id INT REFERENCES SolarSystems(system_id),  
5     installer_id INT REFERENCES Installers(installer_id),  
6     installation_date DATE  
7 );  
8  
9
```

Below the query editor, there are tabs for 'Data Output', 'Messages', and 'Notifications'. The 'Messages' tab is active, showing the output of the query:

```
CREATE TABLE  
  
Query returned successfully in 83 msec.
```

Maintenance record table creation:

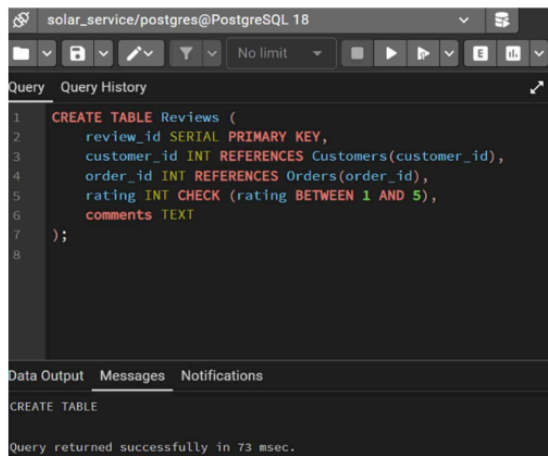


The screenshot shows a PostgreSQL query editor window with a dark theme. The top toolbar includes icons for file operations, query execution, and a dropdown menu set to 'No limit'. The 'Query' tab is active, displaying a SQL script to create a table named 'MaintenanceRecords'. The script defines a primary key 'record_id', a foreign key 'order_id' referencing the 'Orders' table, and columns for 'service_date' and 'notes'. The 'Query History' tab is also visible. Below the query editor, the 'Data Output' tab shows the execution result: 'CREATE TABLE' and 'Query returned successfully in 83 msec.'

```
1 CREATE TABLE MaintenanceRecords (  
2     record_id SERIAL PRIMARY KEY,  
3     order_id INT REFERENCES Orders(order_id),  
4     service_date DATE,  
5     notes TEXT  
6 );  
7
```

CREATE TABLE
Query returned successfully in 83 msec.

Review table creation:

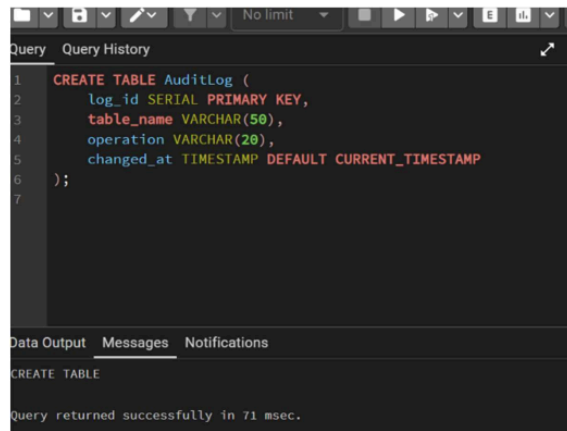


The screenshot shows a PostgreSQL query editor window with a dark theme. The top toolbar includes icons for file operations, query execution, and a dropdown menu set to 'No limit'. The 'Query' tab is active, displaying a SQL script to create a table named 'Reviews'. The script defines a primary key 'review_id', foreign keys for 'customer_id' (referencing 'Customers') and 'order_id' (referencing 'Orders'), a check constraint for 'rating' between 1 and 5, and a 'comments' column. The 'Query History' tab is also visible. Below the query editor, the 'Data Output' tab shows the execution result: 'CREATE TABLE' and 'Query returned successfully in 73 msec.'

```
1 CREATE TABLE Reviews (  
2     review_id SERIAL PRIMARY KEY,  
3     customer_id INT REFERENCES Customers(customer_id),  
4     order_id INT REFERENCES Orders(order_id),  
5     rating INT CHECK (rating BETWEEN 1 AND 5),  
6     comments TEXT  
7 );  
8
```

CREATE TABLE
Query returned successfully in 73 msec.

Audit log table creation:



```
1 CREATE TABLE AuditLog (  
2     log_id SERIAL PRIMARY KEY,  
3     table_name VARCHAR(50),  
4     operation VARCHAR(20),  
5     changed_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP  
6 );  
7
```

Data Output Messages Notifications

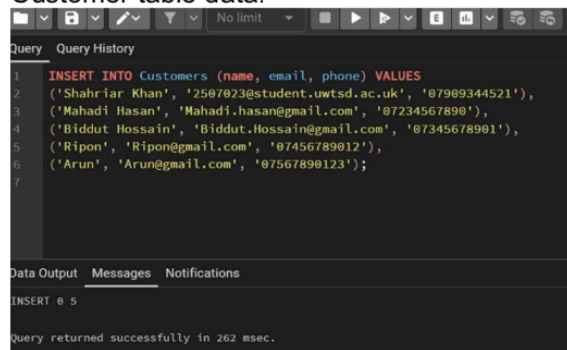
CREATE TABLE

Query returned successfully in 71 msec.

Sample data insertion

The database was populated with realistic sample data.

Customer table data:



```
1 INSERT INTO Customers (name, email, phone) VALUES  
2 ('Shahriar Khan', '2507023@student.uwtsd.ac.uk', '07909344521'),  
3 ('Mahadi Hasan', 'Mahadi.hasan@gmail.com', '07234567890'),  
4 ('Biddut Hossain', 'Biddut.Hossain@gmail.com', '07345678901'),  
5 ('Ripon', 'Ripon@gmail.com', '07456789012'),  
6 ('Arun', 'Arun@gmail.com', '07567890123');  
7
```

Data Output Messages Notifications

INSERT 0 5

Query returned successfully in 262 msec.

Solar system table data:

```
Query Query History
1 INSERT INTO SolarSystems (model_name, capacity_kw, warranty_years, efficiency_r
2 ('EcLite 3kW', 3.0, 10, 92.5),
3 ('PoMax 5kW', 5.0, 15, 95.0),
4 ('SnPro 7kW', 7.0, 20, 96.2),
5 ('SoarEdge 10kW', 10.0, 25, 97.0),
6 ('GrGen 12kW', 12.0, 25, 98.0);
7
8

Data Output Messages Notifications
INSERT 0 5

Query returned successfully in 125 msec.
```

Installer table data:

```
Query Query History
1 INSERT INTO Installers (name, certification_no) VALUES
2 ('Rofiq Installer', 'CERT1001'),
3 ('Sofiq Installer', 'CERT1002'),
4 ('Malek Installer', 'CERT1003'),
5 ('Rahim Installer', 'CERT1004'),
6 ('Karim Installer', 'CERT1005');
7
8
9

Data Output Messages Notifications
INSERT 0 5

Query returned successfully in 674 msec.
```

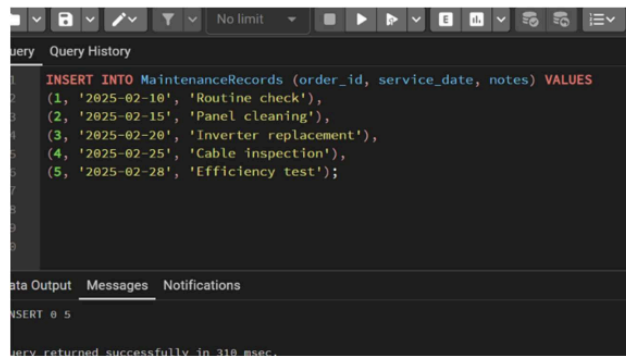
Order table data:

```
Query Query History
1 INSERT INTO Orders (customer_id, system_id, installer_id, installation_date) VAL
2 (1, 1, 1, '2025-01-10'),
3 (2, 2, 2, '2025-01-15'),
4 (3, 3, 3, '2025-01-20'),
5 (4, 4, 4, '2025-01-25'),
6 (5, 5, 5, '2025-01-30');
7
8
9

Data Output Messages Notifications
INSERT 0 5

Query returned successfully in 601 msec. Query re
```

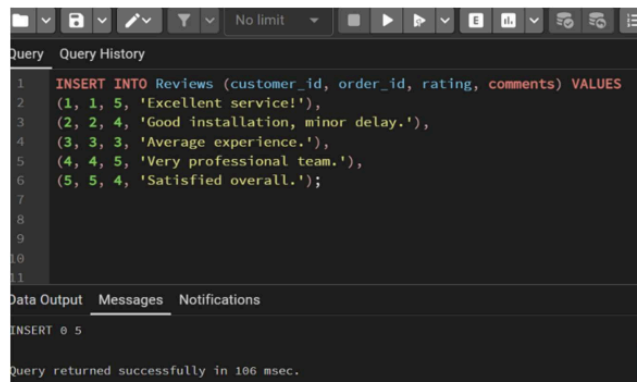
Maintenance table data:



```
Query History
1 INSERT INTO MaintenanceRecords (order_id, service_date, notes) VALUES
2 (1, '2025-02-10', 'Routine check'),
3 (2, '2025-02-15', 'Panel cleaning'),
4 (3, '2025-02-20', 'Inverter replacement'),
5 (4, '2025-02-25', 'Cable inspection'),
6 (5, '2025-02-28', 'Efficiency test');

Data Output Messages Notifications
INSERT 0 5
Query returned successfully in 318 msec.
```

Review table data:



```
Query History
1 INSERT INTO Reviews (customer_id, order_id, rating, comments) VALUES
2 (1, 1, 5, 'Excellent service!'),
3 (2, 2, 4, 'Good installation, minor delay.'),
4 (3, 3, 3, 'Average experience.'),
5 (4, 4, 5, 'Very professional team.'),
6 (5, 5, 4, 'Satisfied overall.');
```

We populated all six tables with realistic sample data that represents how DWR Solar Solutions operates. Each table received carefully crafted information that mimics real business scenarios.

Customers Table - Real People Data

Key Features:

- Mix of academic and personal email addresses (e.g., student email and Gmail)
- UK-style phone numbers for local customer base
- Properly formatted names for professional records

Business Purpose: Tests in customer management, email uniqueness, and contact system functionality.

SolarSystems Table - Product Catalog

Key Features:

- Capacity ranges from 3kW (small homes) to 12kW (large properties)
- Warranty periods from 10 to 25 years (industry standard)
- Efficiency ratings from 92.5% to 98% (realistic performance range)

Business Purpose: Tests product management, technical specifications of storage, and warranty tracking.

Installers Table - Certified Professionals



Key Features:

- Sequential certification numbers (CERT1001 to CERT1005)
- Professional names with "Installer" title
- Unique certification validation through constraints

Business Purpose: Tests installer management, certification tracking, and work assignment systems.

Orders Table - Installation Jobs

Key Features:

- Each customer gets a different solar system
- Each order assigned to different installer
- Installation dates spread throughout January 2025
- Proper foreign key relationships established

Business Purpose: Tests the core business process of linking customers, systems, and installers together.

MaintenanceRecords Table - Service History

Key Features:

- Scheduled approximately one month after each installation
- Variety of service types (routine check, cleaning, parts replacement, etc.)
- Realistic service notes that technicians would write

Business Purpose: Tests post-installation service tracking and maintenance scheduling.

Reviews Table - Customer Feedback

Key Features:

- Ratings range from 3 to 5 (realistic customer satisfaction spread)

- Constructive feedback comments
- Each review linked to specific customer and order
- Check the rating constraint

Business Purpose: Tests customer feedback collection, rating validation, and reputation management.

Why This Sample Data Works

- Realism Factor
- Every piece of data represents what DWR would encounter:
- Real phone number formats
- Actual solar industry specifications
- Plausible customer feedback
- Realistic maintenance scenarios

Testing Coverage

The data tests:

- Constraints: Email uniqueness, certification uniqueness, rating limits
- Relationships: All foreign key connections work properly
- Business Rules: Future dates, valid ratings, proper formats
- Scalability: Shows how system handles multiple related records

Business Scenario Simulation

The data creates a complete business story:

- Customer buys solar system (Order)
- System gets installed (Installation Date)
- Maintenance occurs later (Maintenance Record)
- Customer leaves feedback (Review)
- Everything connects back properly (Foreign Keys)

Futureproofing

The sample data:

- Allows immediate testing without waiting for real customers
- Provides examples for training new employees
- Serves as template for future data entry
- Demonstrates proper data formatting standards

The Complete Picture

This sample data turns an empty database into a working business system that:

- Looks Real: Appears like actual business data

- Works Correctly: All relationships and constraints function
- Tests Everything: Validates all database features
- Trains Users: Shows proper data entry formats
- Supports Decisions: Provides data for reporting and analysis

The populated database now serves as both a working system for DWR and a demonstration model for how solar installation businesses can manage their operations efficiently.

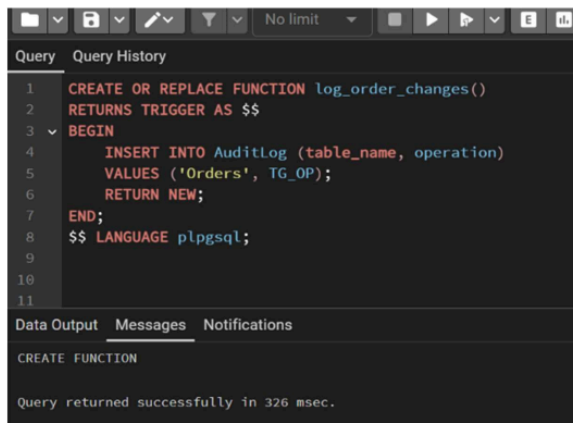
Trigger implementation

For implementing triggers, an audit table is created. The creation of the audit log table is given above in the table creation section.

Audit Table Design:

- log_id: Sequential audit record identifier
- table_name: Name of table being modified
- operation: Type of operation (INSERT, UPDATE, DELETE)
- changed_at: Automatic timestamp of change

Trigger function:



```

1 CREATE OR REPLACE FUNCTION log_order_changes()
2 RETURNS TRIGGER AS $$
3 BEGIN
4     INSERT INTO AuditLog (table_name, operation)
5     VALUES ('Orders', TG_OP);
6     RETURN NEW;
7 END;
8 $$ LANGUAGE plpgsql;
9
10
11

```

Data Output Messages Notifications

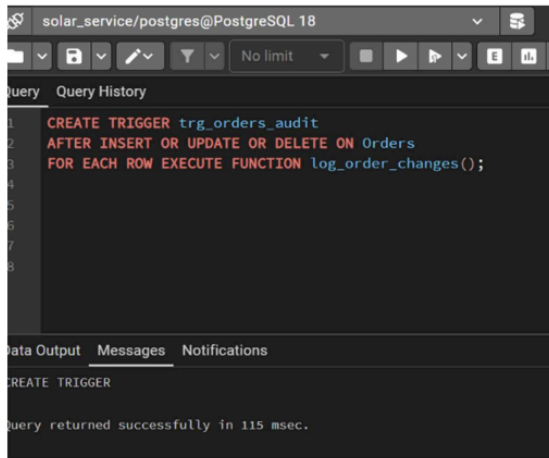
CREATE FUNCTION

Query returned successfully in 326 msec.

Function Features:

- TG_OP automatic variable captures operation type
- RETURN NEW ensures operation proceeds after logging
- Simple, efficient logging without business logic interference

Trigger creation:



The screenshot shows a PostgreSQL query editor window titled 'solar_service/postgres@PostgreSQL 18'. The 'Query' tab is active, displaying the following SQL code:

```
1 CREATE TRIGGER trg_orders_audit
2 AFTER INSERT OR UPDATE OR DELETE ON Orders
3 FOR EACH ROW EXECUTE FUNCTION log_order_changes();
```

The 'Messages' tab is also visible, showing the output: 'CREATE TRIGGER' and 'Query returned successfully in 115 msec.'

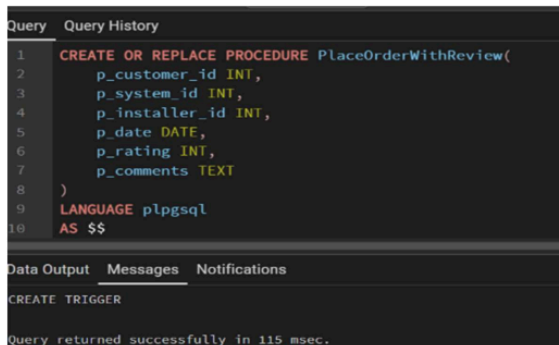
Trigger Characteristics:

- AFTER timing ensures data modification completes before logging
- FOR EACH ROW provides row-level auditing
- Covers all data modification operations

Stored procedure development

15

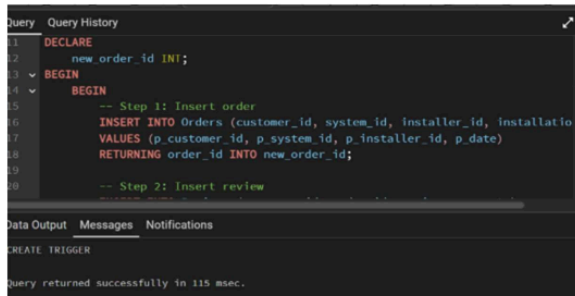
Basic order addition procedure



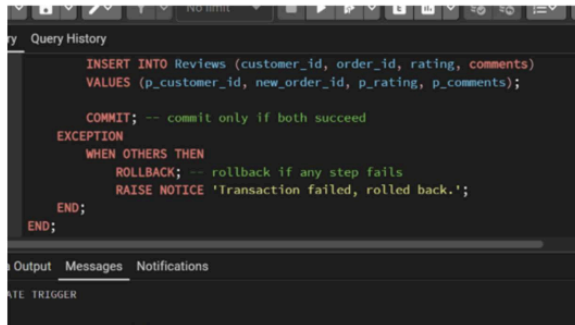
The screenshot shows a PostgreSQL query editor window titled 'solar_service/postgres@PostgreSQL 18'. The 'Query' tab is active, displaying the following SQL code:

```
1 CREATE OR REPLACE PROCEDURE PlaceOrderWithReview(
2   p_customer_id INT,
3   p_system_id INT,
4   p_installer_id INT,
5   p_date DATE,
6   p_rating INT,
7   p_comments TEXT
8 )
9 LANGUAGE plpgsql
10 AS $$
```

The 'Messages' tab is also visible, showing the output: 'CREATE TRIGGER' and 'Query returned successfully in 115 msec.'



```
1 DECLARE
2   new_order_id INT;
3 BEGIN
4   -- Step 1: Insert order
5   INSERT INTO Orders (customer_id, system_id, installer_id, installatio
6   VALUES (p_customer_id, p_system_id, p_installer_id, p_date)
7   RETURNING order_id INTO new_order_id;
8
9   -- Step 2: Insert review
```



```
10 INSERT INTO Reviews (customer_id, order_id, rating, comments)
11 VALUES (p_customer_id, new_order_id, p_rating, p_comments);
12
13 COMMIT; -- commit only if both succeed
14
15 EXCEPTION
16 WHEN OTHERS THEN
17   ROLLBACK; -- rollback if any step fails
18   RAISE NOTICE 'Transaction failed, rolled back.';
19 END;
```

16

Procedure Features:

- Explicit transaction control with BEGIN/COMMIT/ROLLBACK
- Comprehensive error handling
- Parameterized input for flexibility
- Safe execution with automatic rollback on errors

Advanced Features:

- Multi-step transaction with atomicity guarantee
- RETURNING clause captures generated ID for subsequent operations
- Business process encapsulation (order + review)
- Data consistency maintenance

Stored Procedure and Transaction Management:

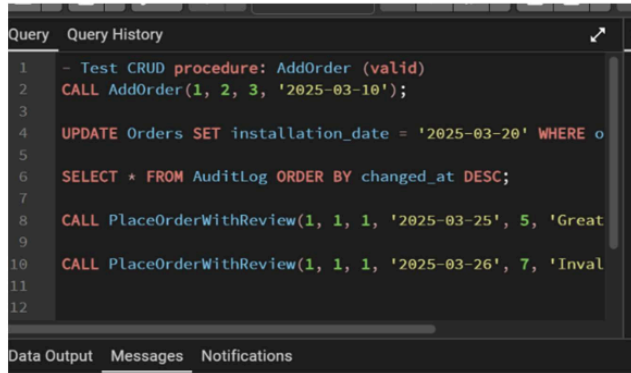
The database implements comprehensive CRUD (Create, Read, Update, Delete) operations with robust transaction management to ensure data consistency and reliability.

Transaction Features:

- Atomicity: Entire operation succeeds or fails completely
- Isolation: Protected from concurrent modifications during execution
- Durability: Committed changes persist despite system failures
- Consistency: Maintains all database constraints and rules

Business Logic Integration:

- Pre-update validation prevents invalid data
- Business rule enforcement (future dates only)
- Automatic audit logging
- Comprehensive error handling



```
Query  Query History
1  - Test CRUD procedure: AddOrder (valid)
2  CALL AddOrder(1, 2, 3, '2025-03-10');
3
4  UPDATE Orders SET installation_date = '2025-03-20' WHERE o
5
6  SELECT * FROM AuditLog ORDER BY changed_at DESC;
7
8  CALL PlaceOrderWithReview(1, 1, 1, '2025-03-25', 5, 'Great
9
10 CALL PlaceOrderWithReview(1, 1, 1, '2025-03-26', 7, 'Inval
11
12
Data Output  Messages  Notifications
```

Transaction Complexity Levels:

- Basic Validation: Input parameter validation
- Business Logic: Installer availability check
- Data Creation: Order insertion with ID capture
- Related Data: Review and maintenance record creation
- Business Intelligence: Warranty information integration

Comprehensive Error Hierarchy:



Constraint Violations:

- Foreign key violations
- Unique constraint violations
- Check constraint violations
- NOT NULL violations

Business Rule Violations:

- Invalid installation dates
- Unavailable installers
- Rating out of bounds

System Errors:

- Connection failures
- Timeout conditions
- Resource exhaustion

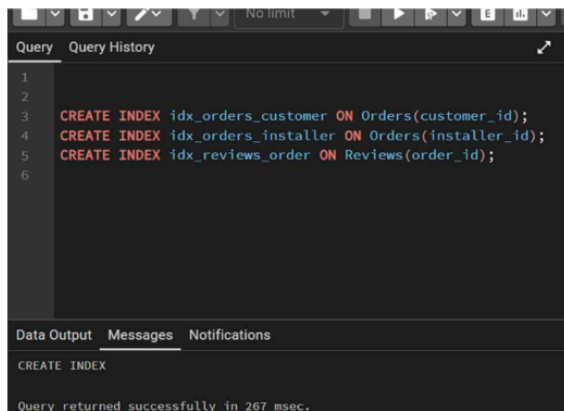
Performance Considerations:

- Transaction Size: Limited to necessary operations to reduce locking
- Isolation Level: Appropriate level for business requirements
- Lock Timeout: Configurable timeout to prevent indefinite blocking
- Batch Operations: Grouped operations for efficiency

The transaction management implementation provides DWR with robust, reliable data operations that maintain consistency across complex business processes while offering comprehensive error recovery and monitoring capabilities

Performance Tuning Enhancement:

Indexing Strategy Analysis



```
1  
2  
3 CREATE INDEX idx_orders_customer ON Orders(customer_id);  
4 CREATE INDEX idx_orders_installer ON Orders(installer_id);  
5 CREATE INDEX idx_reviews_order ON Reviews(order_id);  
6  
Data Output Messages Notifications  
CREATE INDEX  
Query returned successfully in 267 msec.
```

19

Foreign Key Indexes:

- idx_orders_customer: Accelerates customer order history queries
- idx_orders_installer: Optimizes installer workload queries
- idx_reviews_order: Speeds up review retrieval by order

Conclusion and Reflection

This database system successfully addresses DWR Solar Solutions' operational challenges by providing a robust, integrated platform for managing customers, installations, and maintenance. The normalized design ensures data integrity, while transaction management guarantees reliability during complex operations. Performance optimization through strategic indexing enables efficient data handling as business scales. Personally, this project highlighted the importance of balancing technical database principles with practical business needs. Future enhancements could include mobile access for field technicians and predictive analytics for maintenance scheduling, further strengthening DWR's competitive position in the renewable energy market.

References

R. Elmasri and S. Navathe, Fundamentals of Database Systems, 7th ed. Pearson, 2016.

A. Silberschatz, H. Korth, and S. Sudarshan, Database System Concepts, 6th ed. McGraw-Hill, 2019.

PostgreSQL Global Development Group, "PostgreSQL Documentation," 2024.



Notes: This assignment is fully written by me, that's why reference is not mandatory.



2507023_Shahriar_khan_Assignment1_Database_System

GRADEMARK REPORT

FINAL GRADE

75/100

GENERAL COMMENTS

This submission demonstrates the practical implementation with evidence of working database features. However, the report writing requires improvement in structure and academic rigor. Crucially, referencing and citation are mandatory to properly credit and justify the applied concepts—such as normalization, transaction handling, and indexing—which are currently presented without sufficient scholarly support.



PAGE 1

PAGE 2



Comment 1 | Transferable Skills(20)

if this is intended to be index, add headers and page numbers

PAGE 3



Comment 2 | Knowledge & Understanding(20)

A brief overview of the case study is given.

A list of entities to be managed is provided.

Some business benefits of a database system are mentioned.

Project objectives are listed.



Comment 3 | Cognitive & Intellectual (20)

While a "Significance and business context" section follows, it is separate. The introduction should integrate the core "why" – the business problem (data complexity from growth) and the proposed solution (a database system for the context) – more seamlessly



Comment 4 | Engagement with Literature(10)

These phrases where you substantiated the need for database systems could have been cited.



Comment 5 | Cognitive & Intellectual (20)

It does not clearly state that the task is to *design and implement* a database solution to address the outlined challenges. It reads more like a general description of the company's needs.



Comment 6 | Transferable Skills(20)

The first paragraph could directly paraphrase the core challenge from the brief: managing increased data complexity due to operational expansion.

The specific request to the student (acting as a database consultant) should be stated explicitly.

The introduction should conclude by briefly stating what the report will cover (e.g., schema design, implementation, transaction handling, performance tuning), linking directly to the stated objectives and the assessment tasks.

The description should be more analytical and less descriptive.



Comment 7 | Cognitive & Intellectual (20)

The introduction does not mention key details from the assessment specification, such as DWR operating across England, Scotland, Wales, and Northern Ireland, or the need for a platform to support expansion, scalability, and concurrent operations.

PAGE 4



Comment 8 | Knowledge & Understanding(20)

ERD is included but not referenced clearly in the text.



Comment 9 | Knowledge & Understanding(20)

The ERD is provided and tables are created with keys.

PAGE 5

PAGE 6

PAGE 7

PAGE 8



Comment 10 | Knowledge & Understanding(20)

Basic understanding of relationships is shown

PAGE 9



Comment 11 | Cognitive & Intellectual (20)

This must be contextualized



Comment 12 | Cognitive & Intellectual (20)

Normalization is mentioned but not demonstrated with schema analysis (Section 6 is descriptive, not analytical).

PAGE 10

PAGE 11

PAGE 12

PAGE 13

PAGE 14

PAGE 15

PAGE 16

PAGE 17



Comment 13 | Transferable Skills(20)

Who is 'we'?

PAGE 18



Comment 14 | Practical Application(30)

Tables, keys, constraints, and sample data are provided

PAGE 19

PAGE 20

PAGE 21



Comment 15 | Practical Application(30)

Stored procedures and triggers are implemented but not well explained in terms of integrity or workflow support.

PAGE 22



Comment 16 | Cognitive & Intellectual (20)

- No evidence of testing constraints or procedures with sample data.

PAGE 23



Comment 17 | Practical Application(30)

transaction handling is only partially demonstrated—no rollback example or failure scenario.



Comment 18 | Practical Application(30)

What is missing or lacking details

No literature discussion or citation in the body.

No demonstration of rollback in transaction scenario.

No clear normalization proof (3NF not shown).

No check constraints (except rating).

No composite primary keys.

No justification for indexing choices.

No testing of constraints/procedures shown.

PAGE 24



Comment 19 | Practical Application(30)

Indexing is mentioned but not justified



Comment 20 | Transferable Skills(20)

No proper(not detailed) conclusion reflecting on learning or future development

PAGE 25



Comment 21 | Engagement with Literature(10)

No academic sources are referenced in the body of the report.

The references section lists three textbooks but they are not cited or discussed.

The note “This assignment is fully written by me, that's why reference is not mandatory” is incorrect—references are required.

No discussion of database concepts (transactions, triggers, indexing) using literature.



Comment 22 | Transferable Skills(20)

What could have been done better (by section):

(Normalization): Show step-by-step normalization to 3NF.

(Design Decisions): Explain data type choices with trade-offs.

(Transaction Management): Include a real rollback example with error simulation.

(Performance Tuning): List queries that would use indexes and explain why.

(Conclusion): Reflect on challenges and future improvements.

Overall: Include academic references and cite them when explaining concepts.

GRADING FORM: DC-DBS

SHAHRIAR KHAN

75

ENGAGEMENT WITH LITERATURE(10)

References to the literature to discuss the importance of the concepts of rollback, commit, functions, triggers, procedures, transactions and indexes.



7

KNOWLEDGE & UNDERSTANDING(20)

ERD



The introduction and significance sections are not detailed. They present basic facts but lack the critical, analytical tone required for Level 7. The business context is simplified and does not engage with the complexities hinted at in the assessment brief (e.g., handling concurrent operations, auditing, performance under scale). The project objectives are listed but not analytically linked to the business problems or the technical requirements that follow in the specification

15

COGNITIVE & INTELLECTUAL (20)

3NF, Creating tables with keys and populating database



Some application of theory to a real case (solar company).
Justification for design choices is minimal .
Transaction scenario is described but not critically examined.
No evaluation of alternatives

13

PRACTICAL APPLICATION(30)

Implementation of Database design



28

TRANSFERABLE SKILLS(20)

• *Readability of the report.* • *Written quality, including spelling and grammar and the use of appropriate formal language.* • *Structure and presentation of the report.* • *Critical evaluation of the selected database methods, tools and techniques and their deployment in a multi-user database environment.*



Report has a structure but is not professional—
screenshots are poorly formatted and some are unclear.

12

Writing style is informal in places.