A.

This comprehensive report dives deep into staff members' sales performance, with a primary focus on delivering valuable insights into both individual sales and the overall sales performance of individual staff. By closely examining the monetary impact that each staff member brings to the company, this report aims to not only provide a comprehensive overview of their sales but also serve as a strategic tool for enhancing processes related to promotions and training initiatives. This report serves the business by providing an understanding of their staff's sales dynamics. It goes beyond the surface, aiming to uncover the specific individual contributions and their cumulative effect on the company's overall performance. The strategic significance of this report lies in its potential to shape promotional strategies within the organization. Through a detailed analysis of individual sales performances, businesses can identify top staff members deserving of recognition and advancement. By aligning promotions and training initiatives with the monetary gains brought in by staff, the business can ensure that their investments yield evident returns. This approach fosters a culture of recognition and growth, where staff members are motivated to excel, knowing that their contributions are not only acknowledged but also directly tied to their professional development.

A1. The essential tables for this report include the payment and staff tables. From the staff table we will use the staff_id, first_name, and last_name fields. From the payment table we will use the payment_id, payment_date, and amount fields where the staff_id (Foreign Key) matches the staff_id (Primary Key) from the staff table. All the above fields were used in the creation of the detailed table with a single transformation concatenating the first_name and last_name fields into the full_name field. The summary table will use the transformed field full_name, from the detailed table. The summary table will also use the sum of the amount field from the detailed table to create the total_sales field.

A2. The detailed table will include the following fields with their definitions.

staff_id (Integer) Unique identifier for staff members.

full_name (Text) Concatenation of first_name and last_name for a comprehensive representation.

payment_id (Integer) Unique identifier for payment transactions.

payment_date (Timestamp) Date and time of the payment transaction.

amount (Numeric) The value of the payment transaction.

The summary table will consist of the following fields with their definitions.

full_name (Text) Concatenation of first name and last name for a comprehensive representation.

total_sales (Numeric) The total revenue generated by each staff member.

A3. The primary tables utilized are the staff and payment tables.

A4. The transformation of first_name and last_name into a full_name field is a strategic decision, enhancing usability, consistency, and efficiency in data retrieval and communication. This approach ensures a more comprehensive and accessible representation of staff member names.

A5. The detailed table supports transaction-level analysis, allowing businesses to delve into individual staff members' transactions. This aids in understanding peak payment periods, customer payment behavior, and areas for improvement in payment processing. Additionally, it identifies top-performing staff members, enabling tailored training or incentives based on transaction data. The summary table provides a high-level overview of total revenue generated by each staff member, facilitating performance assessments, incentive structures, and acknowledgment of top contributors. It offers a quick financial overview by summarizing total revenue across all staff members, providing a snapshot of overall business financial performance.

A6. To maintain relevance, the report should be refreshed monthly. This frequency aligns with business cycles, financial reporting standards, and decision-making needs, striking a balance between timeliness and stability for effective operational and strategic management.

B. Original code for the function performing the transformation.

```
CREATE OR REPLACE FUNCTION concat_names(first_name text, last_name text)

RETURNS text AS $$

BEGIN

 RETURN first_name || ' ' || last_name;

END;

$$ LANGUAGE plpgsql;
```

C. SQL code to create detailed and summary tables.

```
CREATE TABLE detailed_table(

   staff_id INT,

   full_name TEXT,

   payment_id INT,
    payment_date TIMESTAMP,
    amount NUMERIC);

CREATE TABLE summary_table AS
SELECT
    full_name,
    SUM(amount) AS total_sales
FROM
    detailed_table
GROUP BY
     full_name;
```

D. SQL query to extract raw data for the detailed section.

```
INSERT INTO detailed_table (
 staff_id,
 full_name,
 payment_id,
 payment_date,
 amount
 )
 SELECT
 s.staff_id,
 concat_names(s.first_name, s.last_name) AS full_name,
 p.payment_id,
 p.payment_date,
 p.amount
 FROM
 staff s
 JOIN
 payment p ON s.staff_id = p.staff_id;
```

E. SQL code to create a trigger for updating the summary table.

```
CREATE OR REPLACE FUNCTION update_summary_trigger_function()
RETURNS TRIGGER AS $$
BEGIN
DELETE FROM summary_table
WHERE full_name = NEW.full_name;
 INSERT INTO summary_table (
 full_name,
 total_sales
 )
 SELECT
 full_name,
 SUM(amount) AS total_sales
 FROM
 detailed_table
 WHERE
 full_name = NEW.full_name
 GROUP BY
 full_name;
 RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER update_summary_trigger
AFTER INSERT ON detailed_table
FOR EACH ROW
EXECUTE FUNCTION update_summary_trigger_function();
```

F. SQL stored procedure to refresh detailed and summary tables.

```
CREATE OR REPLACE PROCEDURE refresh_tables()
LANGUAGE plpgsql
AS $$
BEGIN
 DELETE FROM detailed_table;
 INSERT INTO detailed_table (
 staff_id,
 full_name,
 payment_id,
 payment_date,
 amount
 )
 SELECT
 s.staff_id,
 concat_names(s.first_name, s.last_name) AS full_name,
 p.payment_id,
 p.payment_date,
 p.amount
 FROM
 staff s
 JOIN
 payment p ON s.staff_id = p.staff_id;
 DELETE FROM summary_table;
 INSERT INTO summary_table (
 full_name,
 total_sales
 )
 SELECT
 full_name,
 SUM(amount) AS total_sales
 FROM
 detailed_table
 GROUP BY
 full_name;
END;
$$;
```

F1. PostgreSQL provides a powerful job scheduling tool called pg_cron, it enables users to automate the execution of tasks at specified intervals or in response to events within a PostgreSQL database. It is specifically designed for use with the plpgsql language, making it a suitable choice for scheduling and automating tasks in a PostgreSQL environment.