Hayley Wilkinson D191 PA

**Business question:** How many rentals are made each month that are unpaid to date and what is the average repayment time for those that have been paid?

**Section A:**

1. In the report I used the rental data (rental id, staff id, and rental date) in order to generate the list of rentals, date of rental and the staff associated. The staff id was then used to cross reference the staff id to the store id, to allow us to examine the data by location. The payment date from the payment data was matched using the rental id from the rental data in order to identify which rentals were paid or unpaid as well as calculate the number of days between rental and payment which would late be averaged.
2. The rentals table, staff table and payment table were necessary in order to generate the detailed and summary sections of the report.
3. In the detailed report the rental id, customer id, rental date, staff id, store id, payment date and days between rental and payment dates are the included fields. The summary report includes the year and month of rental, the store id, the number of rentals made that month, the number of rentals made that month that remain unpaid to as of the current date, and the average number of days between rental and payment dates excluding those that remain unpaid.
4. The rental date timestamp was transformed to YYYY-MM on the summary report in order to group and present the monthly rental data concisely.
5. The detailed report can be used to analyze the rental and payment data including the accounts that remain unpaid to date. Analysis of this report could include identifying accounts that are not only past due but if there are multiple instances of nonpayment or late payment for that account at one or more store locations. The summary report can be utilized to analyze the business activity of both locations and the number of unpaid accounts and average time it takes to collect payment on rentals. This allows for trends in level of activity as well as any trend in nonpayment or late payment at each location. These reports can also be utilized when reviewing payment, late fee and collection policies as it demonstrates that most rentals remain unpaid for over 600 days before payment is collected.
6. The report should be refreshed at least monthly in order to capture the complete data for the month for stakeholders unless their communication preferences are more frequent, which would indicate refreshing the report as often as required to meet those preferences.

**Sections B-E:** All code is included at then end of the document

**Section F:**

1. The stored procedure can be run on a schedule using an external task scheduler or an extension that allows task scheduling. The task could be created to run the stored procedure on a monthly basis or more often to ensure the data freshness.

**Section H-I:** No web sources were used to acquire data or segments of third-party code, nor were any sources utilized for quotes, paraphrasing or summarizing for this submission.

**SECTION B-E CODE:**

DROP TABLE IF EXISTS rental\_payment;

CREATE TABLE IF NOT EXISTS rental\_payment (

rental\_id INT,

customer\_id INT,

staff\_id INT,

store\_id INT,

rental\_date DATE,

payment\_date DATE,

days\_between NUMERIC

);

SELECT \*

FROM rental\_payment;

INSERT INTO rental\_payment

SELECT rental.rental\_id, rental.customer\_id, rental.staff\_id, staff.store\_id,

DATE(rental.rental\_date),

DATE(payment.payment\_date),

DATE\_PART('day', payment.payment\_date - rental.rental\_date) AS days\_between

FROM rental

LEFT JOIN staff ON rental.staff\_id = staff.staff\_id

LEFT JOIN payment ON rental.rental\_id = payment.rental\_id

ORDER BY rental\_id;

SELECT \*

FROM rental\_payment;

DROP TABLE IF EXISTS payment\_lag;

CREATE TABLE IF NOT EXISTS payment\_lag(

store\_id INT,

  year\_month TEXT,

months\_new\_rentals BIGINT,

unpaid\_rentals BIGINT,

avg\_days\_between\_rental\_payment NUMERIC

);

SELECT \*

FROM payment\_lag;

INSERT INTO payment\_lag

SELECT rental\_payment.store\_id AS store\_id,

to\_char(rental\_date, 'YYYY-MM') as year\_month,

COUNT(rental\_id) AS months\_new\_rentals,

COUNT(rental\_id) FILTER (WHERE payment\_date IS NULL) AS unpaid\_rentals,

ROUND(AVG(days\_between)) AS avg\_days\_between\_rental\_payment

FROM rental\_payment

GROUP BY rental\_payment.store\_id, rental\_date

ORDER BY rental\_date, rental\_payment.store\_id;

SELECT \*

FROM payment\_lag;

CREATE OR REPLACE FUNCTION rental\_date\_update()

RETURNS TRIGGER

LANGUAGE PLPGSQL

AS

$$

BEGIN

IF NEW.rental\_date <> OLD.rental\_date THEN

INSERT INTO payment\_lag (year\_month)

VALUES (NEW.rental\_date);

END IF;

RETURN NEW;

END;

$$;

CREATE TRIGGER rental\_date\_update

AFTER UPDATE

ON rental\_payment

FOR EACH ROW

EXECUTE PROCEDURE rental\_date\_update();

CREATE OR REPLACE PROCEDURE refresh\_rp ()

LANGUAGE PLPGSQL

AS $$

BEGIN

TRUNCATE TABLE rental\_payment;

INSERT INTO rental\_payment

SELECT rental.rental\_id, rental.customer\_id, rental.staff\_id, staff.store\_id,

DATE(rental.rental\_date),

DATE(payment.payment\_date),

DATE\_PART('day', payment.payment\_date - rental.rental\_date) AS days\_between

FROM rental

LEFT JOIN staff ON rental.staff\_id = staff.staff\_id

LEFT JOIN payment ON rental.rental\_id = payment.rental\_id

ORDER BY rental\_id;

END;$$;

CALL refresh\_rp;

CREATE OR REPLACE PROCEDURE refresh\_pl()

LANGUAGE PLPGSQL

AS $$

BEGIN

TRUNCATE TABLE payment\_lag;

INSERT INTO payment\_lag

SELECT rental\_payment.store\_id AS store\_id,

to\_char(rental\_date, 'YYYY-MM') as year\_month,

COUNT(rental\_id) AS months\_new\_rentals,

COUNT(rental\_id) FILTER (WHERE payment\_date IS NULL) AS unpaid\_rentals,

ROUND(AVG(days\_between)) AS avg\_days\_between\_rental\_payment

FROM rental\_payment

GROUP BY rental\_payment.store\_id, rental\_date

ORDER BY rental\_date, rental\_payment.store\_id;

END;$$;

CALL refresh\_pl();

CREATE OR REPLACE PROCEDURE refresh\_rp\_pl()

LANGUAGE PLPGSQL

AS $$

BEGIN

CALL refresh\_rp();

CALL refresh\_pl();

END;$$;

COMMENT ON PROCEDURE public.refresh\_rp\_pl()

    IS '‘should be run at least monthly to capture all new data and updates to previously available data';

CALL refresh\_rp\_pl();

**PANOPTO LINK:**

<https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=84b8516a-9290-4507-a406-afa1011a430f>