A. Summarize one real-world business report that can be created from the attached Data Sets and Associated Dictionaries.

1. Describe the data used for the report.
    - "Data around the product of DVD's and the various types of metadata and classification data that one might use in management of that product. Also contained in the database is standard business fact tables such as inventory, payment records, and Staffing"
2. Identify two or more specific tables from the given dataset that will provide the data necessary for the detailed and the summary sections of the report.
    - `Payment`, `Store`, `Staff`, `Address`
3. Identify the specific fields that will be included in the detailed and the summary sections of the report.
    - `Payment`: payment_date, staff_id, amount
    - `Staff`: staff_id, store_id
    - `Store`: store_id, address_id
    - `Address`: address_id, address
4. Identify one field in the detailed section that will require a custom transformation and explain why it should be transformed. For example, you might translate a field with a value of `N` to `No` and `Y` to `Yes`.
    - Will have to cast the date times in the `Payment` table to be just be the Year datepart for the roll up summary by year
    - Casting the `Payment` field "amount" from numeric to money to more accurately reflect the underlying data
5. Explain the different business uses of the detailed and the summary sections of the report.
    - Be able to see the most profitable rental locations
    - See payment revenue per location per year for trended metrics on each locations profitability
6. Explain how frequently your report should be refreshed to remain relevant to stakeholders.
    - Since the revenue per location is calculated live it can be refreshed as needed or live via a trigger calling a stored procedure.
    - To see the most profitable locations or products the definitive results wont finalize until fiscal year's end, but can still be used live for forcasting if desired.

B. Write a SQL code that creates the tables to hold your report sections.

```
CREATE SCHEMA rpt AUTHORIZATION postgres;
COMMENT ON SCHEMA rpt IS 'Reporting';
GRANT ALL PRIVILEGES ON SCHEMA rpt TO postgres;
ALTER USER postgres SET search_path TO rpt, public;

------------------------------------------------------------

CREATE TABLE rpt.report_data
(
    "payment_date" timestamp without time zone NOT NULL,
    "address" character varying (255) NOT NULL,
    "amount" numeric
);
ALTER TABLE rpt.report_data OWNER to postgres;
```

```
-------------------------------------------------------------

CREATE TABLE rpt.report_data_clean
(
    "Year" smallint NOT NULL,
    "Location" character varying (255) NOT NULL,
    "Revenue" money
);
ALTER TABLE rpt.report_data_clean OWNER to postgres;

-------------------------------------------------------------

CREATE TABLE rpt.location_trended
(
    "Year" smallint NOT NULL,
    "Location" character varying (255) NOT NULL,
    "Revenue" money
);
ALTER TABLE rpt.location_trended OWNER to postgres;

CREATE UNIQUE INDEX "CIX_Year_Loc"
    ON rpt.location_trended USING btree
    ("Year" ASC NULLS LAST, "Location" COLLATE pg_catalog."default" ASC
NULLS LAST)
    INCLUDE ("Year", "Location")
    TABLESPACE pg_default;
    ALTER TABLE rpt.location_trended CLUSTER ON "CIX_Year_Loc";

-------------------------------------------------------------

CREATE TABLE rpt.location_top
(
    "Year" smallint NOT NULL,
    "Location" character varying(255) NOT NULL,
    "Revenue" money NOT NULL,
    "Rank" int NOT NULL
);
ALTER TABLE rpt.location_top OWNER to postgres;

CREATE INDEX "CIX_Year_Revenue"
    ON rpt.location_top USING btree
    ("Year" DESC NULLS LAST, "Revenue" DESC NULLS LAST)
    INCLUDE ("Year", "Revenue")

ALTER TABLE rpt.location_top CLUSTER ON "CIX_Year_Revenue";
```

C. Write a SQL query that will extract the raw data needed for the Detailed section of your report from the source database and verify the data`s accuracy.

```
INSERT INTO rpt.report_data
SELECT a.address
     , p.payment_date
     , p.amount
FROM public.payment AS p
    LEFT JOIN public.staff AS s ON s.staff_id = p.staff_id
    LEFT JOIN public.store AS st ON st.store_id = s.store_id
    LEFT JOIN public.address AS a ON a.address_id = st.address_id
;
```

D. Write code for function(s) that perform the transformation(s) you identified in part A4.

```
CREATE OR REPLACE FUNCTION rpt.FN_Clean_Data() RETURNS trigger LANGUAGE
'plpgsql' AS $$
BEGIN
    INSERT INTO rpt.report_data_clean
    SELECT DATE_PART('year', payment_date) AS Year
         , address AS Location
         , CAST(amount AS money) AS Revenue
    FROM rpt.report_data;
    RETURN NULL;
END;
$$;
ALTER FUNCTION rpt.FN_Clean_Data() OWNER TO postgres;
COMMENT ON FUNCTION rpt.FN_Clean_Data()
IS 'New data in rpt.report_data gets cleaned and inserted into
rpt.report_data_clean';

CREATE TRIGGER TR_ETL AFTER INSERT ON rpt.report_data
    FOR STATEMENT
    EXECUTE FUNCTION rpt.FN_Clean_Data();
COMMENT ON TRIGGER TR_ETL ON rpt.report_data
IS 'Update reports when new data is added to the rpt.report_data table';
```

E. Write a SQL code that creates a trigger on the detailed table of the report that will continually update the summary table as data is added to the detailed table.

```
CREATE OR REPLACE FUNCTION rpt.FN_ETL() RETURNS trigger LANGUAGE 'plpgsql'
AS $$
BEGIN
    TRUNCATE TABLE rpt.location_trended;
    INSERT INTO rpt.location_trended
    SELECT "Year", "Location", SUM("Revenue")
    FROM rpt.report_data_clean
```

```
    GROUP BY "Year", "Location";

    TRUNCATE TABLE rpt.location_top;
    INSERT INTO rpt.location_top
    SELECT "Year", "Location", SUM("Revenue")
         , RANK() OVER(ORDER BY "Year" DESC, SUM("Revenue") DESC) AS
"Rank"
    FROM rpt.report_data_clean
    WHERE "Year" IN (
        --   CAST(DATE_PART('year', NOW()) AS INT)     -- Current Year
        -- , CAST(DATE_PART('year', NOW()) AS INT) - 1 -- Prior Year
          (SELECT MAX("Year") FROM rpt.report_data_clean)
        , (SELECT MAX("Year") - 1 FROM rpt.report_data_clean)
    )
    GROUP BY "Year", "Location";
    RETURN NULL;
END
$$;
ALTER FUNCTION rpt.FN_ETL() OWNER TO postgres;
COMMENT ON FUNCTION rpt.FN_ETL()
IS 'New data in rpt.report_data_clean so update all reports';

CREATE TRIGGER TR_Update_Reports AFTER INSERT ON rpt.report_data_clean
FOR STATEMENT
EXECUTE FUNCTION rpt.FN_ETL();
COMMENT ON TRIGGER TR_Update_Reports ON rpt.report_data_clean
IS 'Update reports when new data is added to the rpt.report_data_clean
table';
```

F. Create a stored procedure that can be used to refresh the data in both your detailed and summary tables. The procedure should clear the contents of the detailed and summary tables and perform the ETL load process from part C and include comments that identify how often the stored procedure should be executed.

```


CREATE OR REPLACE FUNCTION rpt."FN_Nuke_From_Orbit"() RETURNS void
LANGUAGE 'plpgsql' AS $$
BEGIN
    TRUNCATE TABLE rpt.report_data;
    TRUNCATE TABLE rpt.report_data_clean;
    TRUNCATE TABLE rpt.location_trended;
    TRUNCATE TABLE rpt.location_top;
END;
$$;
ALTER FUNCTION rpt."FN_Nuke_From_Orbit"() OWNER TO postgres;
COMMENT ON FUNCTION rpt."FN_Nuke_From_Orbit"()
IS 'Wipe all data in the rot schema from the Face of the earth';

CREATE OR REPLACE PROCEDURE rpt."USP_Refresh"() LANGUAGE 'plpgsql' AS $$
```

```
BEGIN
    /*************************************************************
    USAGE:
        This stored procedure can be run at any interval desired.
        Since this performs a full truncate and load of the entire
            rpt schema and all table data this will be expensive
            in resources.
        Recommendation is, depending on geographic factors of
            store locations affecting end of day accounting totals,
            to run the stored procedure in the off hours such as
            at midnight or directly after C.O.B. so the day's
            results are posted immediately after the conclusion
            of that day's business.
    *************************************************************/
    /*************************************************************
    Wipe The entire structure of the rpt schema for a clean
    full rebuild via truncate and load ETL
    *************************************************************/
    PERFORM rpt."FN_Nuke_From_Orbit"();
    /*************************************************************
    Grab all raw data and kick off the live rebuild process
    driven by triggers.
    *************************************************************/
    INSERT INTO rpt.report_data
    SELECT p.payment_date
         , a.address
         , p.amount
    FROM public.payment AS p
        LEFT JOIN public.staff AS S ON s.staff_id = p.staff_id
        LEFT JOIN public.store AS st ON st.store_id = s.store_id
        LEFT JOIN public.address AS a ON a.address_id = st.address_id;
    COMMIT;
END;
$$;
COMMENT ON PROCEDURE rpt."USP_Refresh"()
IS 'Refresh the entire reporting structure';
```

1. Explain how the stored procedure can be run on a schedule to ensure data freshness.
    - The stored procedure can be ran on regularly scheduled intervals via cron or a sql agent to run the stored proc as a job on a schedule.

G. Provide a Panopto video recording that includes a demonstration of the functionality of the code used for the analysis and a summary of the programming environment.

TODO LEFT OFF HERE

Note: For instructions on how to access and use Panopto, use the "Panopto How-To Videos" web link provided below. To access Panopto's website, navigate to the web link titled "Panopto Access," and then choose to log in using the “WGU” option. If prompted, log in using your WGU student portal credentials,

and then it will forward you to Panopto`s website. To submit your recording, upload it to the Panopto drop box titled “XXX.” Once the recording has been uploaded and processed in Panopto's system, retrieve the URL of the recording from Panopto and copy and paste it into the Links option. Upload the remaining task requirements using the Attachments option.

H. Record the web sources you used to acquire data or segments of third-party code to support the application if applicable. Be sure the web sources are reliable.

no code used from outside, only resources to understand syntax, everything written in the VM or written by postgres GUI menu's

I. Acknowledge sources, using in-text citations and references, for content that is quoted, paraphrased, or summarized.

J. Demonstrate professional communication in the content and presentation of your submission.