SAHITHI AREKATLA
U16464865

ADVANCED DATABASE MANAGEMENT
TERM PROJECT REPORT

1. **Modifying the SCD implementation for the dimensional table:**
   Solution:
   Query:

```
CREATE OR REPLACE PROCEDURE update_dim_car()
LANGUAGE plpgsql
AS $$
BEGIN
   -- INSERT event
   INSERT INTO dim_car (featureid, manufacturer, model, engine_volume, airbags,
interior, color, fuel_type, current_flag, effective_timestamp, expired_timestamp)
   SELECT c.featureid, c.manufacturer, m.model_name, c.engine_volume, c.airbags,
i.leather_interior, co.color, fu.fuel_type, TRUE, NOW(), NULL
   FROM features c
   JOIN model m ON m.model_name = c.model
   JOIN interior i ON i.leather_interior = c.leather_interior
   JOIN color co ON co.color = c.color
   JOIN fuel fu ON fu.fuel_type = c.fuel_type
   EXCEPT
   SELECT d.featureid, d.manufacturer, d.model, d.engine_volume, d.airbags, d.interior,
d.color, d.fuel_type, d.current_flag, d.effective_timestamp, d.expired_timestamp
   FROM dim_car d
   WHERE d.current_flag = TRUE;

         -- UPDATE event
UPDATE dim_car SET
   current_flag = true
WHERE
   featureid IN (
      SELECT
         c.carid
      FROM
         features f
         JOIN car c ON f.carid = c.carid
         JOIN model m ON m.modelid = c.modelid
      WHERE
         EXISTS (
            SELECT 1
```

```
            FROM dim_car d
            WHERE
                d.featureid = c.carid
                AND d.manufacturer = c.manufacturer
                AND d.model = m.model_name
                AND (
                    d.engine_volume <> f.engine_volume
                    OR d.airbags <> f.airbags
                    OR d.interior <> f.leather_interior
                    OR d.color <> f.color
                    OR d.fuel_type <> f.fuel_type
                )
                AND d.current_flag = true
        )
    );

INSERT INTO dim_car (featureid, manufacturer, model, engine_volume, airbags,
interior, color, fuel_type, current_flag)
SELECT
    c.carid,
    c.manufacturer,
    m.model_name,
    f.engine_volume,
    f.airbags,
    f.leather_interior,
    f.color,
    f.fuel_type,
    true
FROM
    features f
    JOIN car c ON f.carid = c.carid
    JOIN model m ON m.modelid = c.modelid
WHERE
    (c.carid, c.manufacturer, m.model_name, f.engine_volume, f.airbags, f.leather_interior,
f.color, f.fuel_type, 'Y')
    EXCEPT
    SELECT
        d.featureid, d.manufacturer, d.model, d.engine_volume, d.airbags, d.interior, d.color,
d.fuel_type, d.current_flag
    FROM
        dim_car d
    WHERE
        d.current_flag = true;
```

```
    -- DELETE event
    UPDATE dim_car SET
        current_flag = false,
        expired_timestamp = NOW()
    WHERE
        featureid IN (
            SELECT
                c.carid
            FROM
                features f
                JOIN car c ON f.carid = c.carid
                JOIN model m ON m.modelid = c.modelid
            WHERE
                EXISTS (
                    SELECT 1
                    FROM dim_car d
                    WHERE
                        d.featureid = c.carid
                        AND d.manufacturer = c.manufacturer
                        AND d.model = m.model_name
                        AND d.current_flag = true
                )
        );

    DELETE FROM dim_car d
    USING features f
    WHERE d.featureid = f.featureid
        AND d.current_flag = false;
END;
$$;
```

Query Explanation:
The query creates a stored procedure called update_dim_car. The purpose of this stored procedure is to synchronize data between a features table and a dim_car dimension table. The features table represents the source data, and the dim_car table represents the target data.

The stored procedure handles three events: INSERT, UPDATE, and DELETE using EXCEPT and JOIN commands, to synchronize the data between the source and target tables. The procedure also sets the appropriate current_flag, effective_timestamp, and expired_timestamp values for each row in the dim_car table to ensure that the data is properly versioned.

Output Screenshots:

Insert Command:

INSERT INTO features(ModelID,CarID,YearID,categoryid,interiorid, colorid,fuelid,
gearboxid,mileage, cylinders, airbags,
engine_volume)
values(211, 27, 3,7, 2, 16, 5, 1, 13000, 4,12, 3)
call update_dim_car()

Data Output Messages Notifications

	dim_carid [PK] integer	featureid integer	manufacturer character varying (255)	model character varying (255)	engine_volume integer	airbags integer	interior character varying (255)	color character varying (255)	fuel_type character varying (255)	current_flag boolean	effecti timest
4090	4090	4090	bmw	535	12	3	Yes	White	Hybrid	true	2023-(
4091	4091	4091	hyundai	Elantra	4	2	Yes	White	Diesel	true	2023-(
4092	4092	4092	mercedes-benz	C 230	8	3	Yes	Black	Petrol	true	2023-(
4093	4093	4093	ford	Mustang	6	5	Yes	Blue	Petrol	true	2023-(
4094	4094	4094	honda	CHR	3	12	Yes	Black	Petrol	true	2023-(

Data Output Messages Notifications

	model character varying (255)	engine_volume integer	airbags integer	interior character varying (255)	color character varying (255)	fuel_type character varying (255)	current_flag boolean	effective_timestamp timestamp without time zone	expired_timestamp timestamp without time zone
4090	535	12	3	Yes	White	Hybrid	true	2023-05-03 11:44:54.99321	[null]
4091	Elantra	4	2	Yes	White	Diesel	true	2023-05-03 11:44:54.99321	[null]
4092	C 230	8	3	Yes	Black	Petrol	true	2023-05-03 11:44:54.99321	[null]
4093	Mustang	6	5	Yes	Blue	Petrol	true	2023-05-03 11:44:54.99321	[null]
4094	CHR	3	12	Yes	Black	Petrol	true	2023-05-03 12:23:45.632512	[null]

Update Command:

update features
set engine_volume=20
where featureId=4094
call update_dim_car()

Operators
Procedures
Sequences
Tables (20)
car
car_price
category
color
date_dim
dim_car
dim_gear_

Data Output Messages Notifications

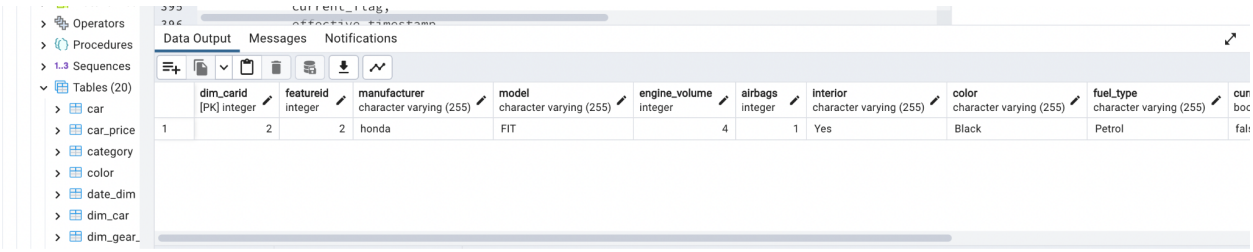
	featureid integer	manufacturer character varying (255)	model character varying (255)	engine_volume integer	airbags integer	interior character varying (255)	color character varying (255)	fuel_type character varying (255)	current_flag boolean
4090	4091	hyundai	Elantra	4	2	Yes	White	Diesel	true
4091	4092	mercedes-benz	C 230	8	3	Yes	Black	Petrol	true
4092	4093	ford	Mustang	6	5	Yes	Blue	Petrol	true
4093	4094	honda	CHR	20	12	Yes	Black	Petrol	false
4094	4094	honda	CHR	3	12	yes	Black	Petrol	true

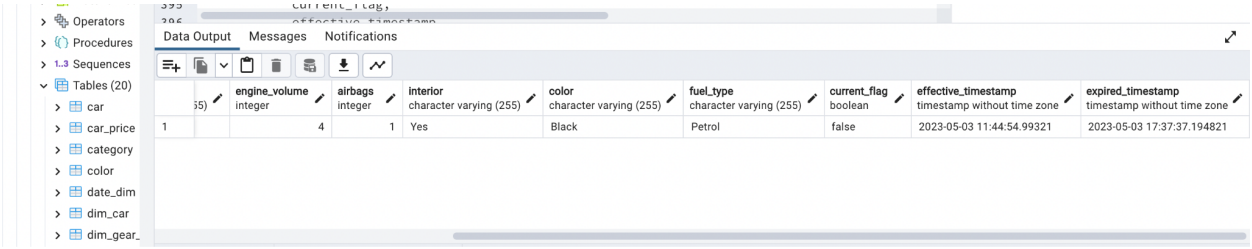
Operators
Procedures
Sequences
Tables (20)
car
car_price
category
color
date_dim
dim_car
dim_gear_

Data Output Messages Notifications

	engine_volume integer	airbags integer	interior character varying (255)	color character varying (255)	fuel_type character varying (255)	current_flag boolean	effective_timestamp timestamp without time zone	expired_timestamp timestamp without time zone
4090	4	2	Yes	White	Diesel	true	2023-05-03 11:44:54.99321	[null]
4091	8	3	Yes	Black	Petrol	true	2023-05-03 11:44:54.99321	[null]
4092	6	5	Yes	Blue	Petrol	true	2023-05-03 11:44:54.99321	[null]
4093	20	12	Yes	Black	Petrol	false	2023-05-03 12:23:45.632512	2023-05-03 17:11:51.742297
4094	3	12	yes	Black	Petrol	true	2023-05-03 17:18:23.059383	[null]

Delete command:
delete from feature where featureid=2
call update_dim_car()





2. **Performance tuning on Fact table:**
   SOLUTION:
   Creating indexes on the foreign key columns will improve the performance of queries that join the fact_car table with the corresponding dimension tables.

   Query:
   CREATE INDEX fact_car_dim_carID_idx ON fact_car(dim_carID);
   CREATE INDEX fact_car_date_iD_idx ON fact_car(date_iD);
   CREATE INDEX fact_car_dim_ProdYeariD_idx ON fact_car(dim_ProdYeariD);
   CREATE INDEX fact_car_dim_gear_boxiD_idx ON fact_car(dim_gear_boxiD);

   Output Screenshots:

```sql
FROM
    dim_car dc join car c on c.manufacturer=dc.manufacturer
    inner join car_price pc on pc.carid = c.carid
    inner join new_car_prices ncp on ncp.carpriceid = pc.carpriceid
    inner join old_car_prices ocp on ocp.carpriceid = pc.carpriceid
    inner JOIN date_dim dd ON dd.date = ncp.date_range
    inner join features f on f.carid = c.carid
     inner join year y on y.yearid = f.yearid
    inner JOIN Dim_gear_box dgb ON dgb.dim_gear_boxid = f.gearboxID
GROUP BY
    dc.dim_carID,
    dd.date_id,
    y.yearid,
    dgb.dim_gear_boxid


drop table fact_car

CREATE INDEX fact_car_dim_carID_idx ON fact_car(dim_carID);
CREATE INDEX fact_car_date_iD_idx ON fact_car(date_iD);
CREATE INDEX fact_car_dim_ProdYeariD_idx ON fact_car(dim_ProdYeariD);
CREATE INDEX fact_car_dim_gear_boxiD_idx ON fact_car(dim_gear_boxiD);
```

Data Output | Messages | Notifications

```
INSERT 0 50000

Query returned successfully in 1 min 47 secs.
```

Total rows: 0 of 0 | Query complete 00:01:47.412 | Ln 151, Col 1

3. **Performing certain data preprocessing techniques in SQL**

Solution:

1. I have used the sql to remove white spaces in the staging tables . I used the trim() function
2. Secondly, due to removing the white spaces in the staging tables they were many duplicates created in the staging tables, so I removed the duplicates also using SQL.

Query:

```sql
-- removing white spaces from sql tables
CREATE OR REPLACE FUNCTION remove_whitespace_from_staging_new_car()
RETURNS VOID AS $$
BEGIN
  UPDATE staging_new_car
  SET
    manufacturer = TRIM(manufacturer),
    model = TRIM(model),
    gear_box_type = TRIM(gear_box_type),
    price_variation = TRIM(price_variation);
END;
$$ LANGUAGE plpgsql;
CALL remove_whitespace_from_staging_new_car();
```

```
CREATE OR REPLACE FUNCTION remove_whitespace_from_staging_old_car()
RETURNS VOID AS $$
BEGIN
  UPDATE staging_old_car
  SET
    Month_Year = TRIM(Month_Year),
    Manufacturer = TRIM(Manufacturer),
    Model = TRIM(Model),
    ProdYear = TRIM(ProdYear);
END;
$$ LANGUAGE plpgsql;
CALL remove_whitespace_from_staging_old_car();



CREATE OR REPLACE FUNCTION remove_whitespace_from_staging_car()
RETURNS VOID AS $$
BEGIN
  UPDATE staging_car
  SET
    Manufacturer = TRIM(Manufacturer),
    Model = TRIM(Model),
    ProdYear = TRIM(ProdYear),
    Category = TRIM(Category),
    Leather_interior = TRIM(Leather_interior),
    Fuel_type = TRIM(Fuel_type),
    Gear_box_type = TRIM(Gear_box_type),
    Color = TRIM(Color);
END;
$$ LANGUAGE plpgsql;
CALL remove_whitespace_from_staging_car();


-- DELETING DUPLICATE VALUES FROM THE STAGING TABLES

-- FROM staging_new_car
DELETE FROM staging_new_car
WHERE (old_price, new_price, date_range,manufacturer, model, prod_year,
gear_box_type, price_variation, price_change) IN (
  SELECT old_price, new_price, date_range,manufacturer, model, prod_year,
gear_box_type, price_variation, price_change
  FROM new_car_prices
```

```
GROUP BY  old_price, new_price, date_range,manufacturer, model, prod_year,
gear_box_type, price_variation, price_change
HAVING COUNT(*) > 1
);
select * from staging_new_car
where old_price=315000.0 and new_price=345000.0 and date_range='2022-04-07
00:00:00' and manufacturer='Hyundai' and model='Accent HCI' and prod_year=2022

-- From old_car_prices

DELETE FROM old_car_prices
WHERE (Month_Year,          Average_Price,          Minimum_Price,
Maximum_price,        Manufacturer,  Model, ProdYear) IN (
SELECT Month_Year,Average_Price,          Minimum_Price,         Maximum_price,
Manufacturer,  Model, ProdYear
FROM old_car_prices
GROUP BY  Month_Year,      Average_Price,          Minimum_Price,
Maximum_price,        Manufacturer,  Model, ProdYear
HAVING COUNT(*) > 1
);
select * from old_car_prices

-- staging_car table
SELECT id, manufacturer, model,prodyear, category, leather_interior, fuel_type,
engine_volume, mileage, color, airbags
FROM Car_prices
GROUP BY id, manufacturer, model,prodyear, category, leather_interior, fuel_type,
engine_volume, mileage, color, airbags
HAVING COUNT(*) > 1
```