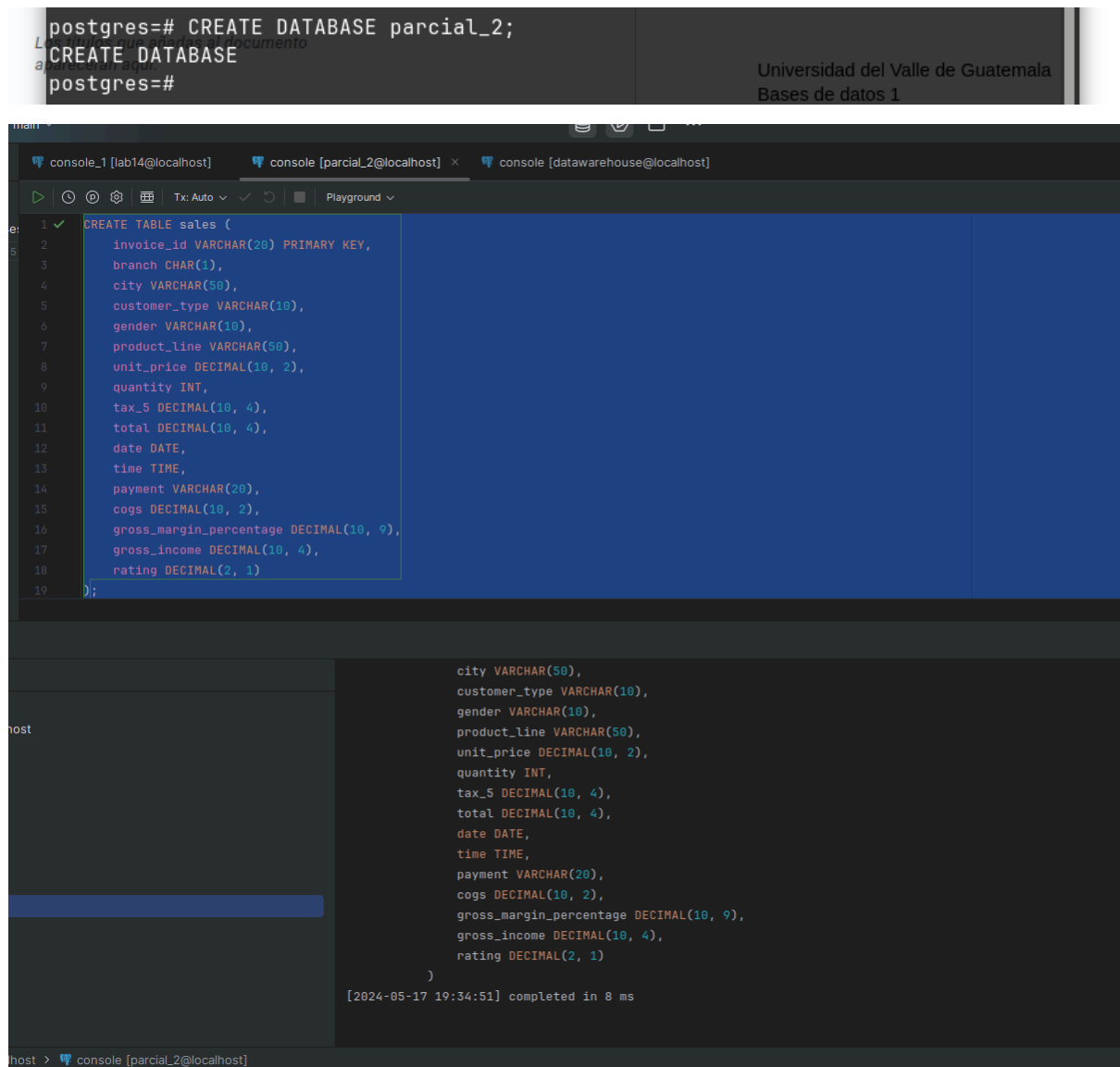


Parcial 2

1. En su instalación local de PostgreSQL cree una base de datos llamada parcial_2 y cree allí la tabla que le servirá para almacenar los datos del archivo CSV.



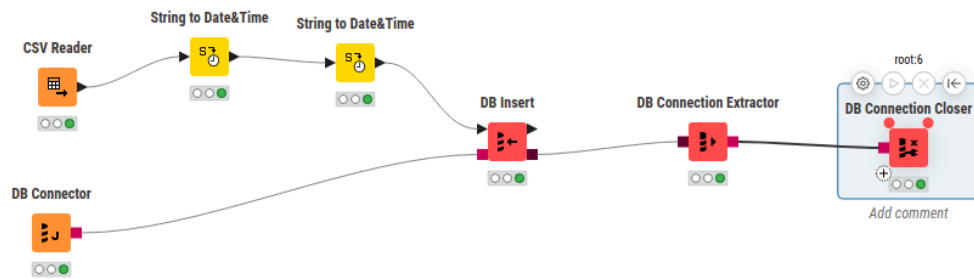
The image shows two screenshots related to PostgreSQL database creation. The top screenshot is a terminal window with the command: `postgres=# CREATE DATABASE parcial_2;` followed by `CREATE DATABASE` and `postgres=#`. The bottom screenshot is a DBeaver console window showing the execution of a SQL script. The script creates a table named `sales` with the following columns: `invoice_id` (VARCHAR(20), PRIMARY KEY), `branch` (CHAR(1)), `city` (VARCHAR(50)), `customer_type` (VARCHAR(10)), `gender` (VARCHAR(10)), `product_line` (VARCHAR(50)), `unit_price` (DECIMAL(10, 2)), `quantity` (INT), `tax_5` (DECIMAL(10, 4)), `total` (DECIMAL(10, 4)), `date` (DATE), `time` (TIME), `payment` (VARCHAR(20)), `cogs` (DECIMAL(10, 2)), `gross_margin_percentage` (DECIMAL(10, 9)), `gross_income` (DECIMAL(10, 4)), and `rating` (DECIMAL(2, 1)). The console output shows the table creation completed successfully on 2024-05-17 at 19:34:51 in 8 ms.

```
postgres=# CREATE DATABASE parcial_2;
CREATE DATABASE
postgres=#
```

```
1 CREATE TABLE sales (
2   invoice_id VARCHAR(20) PRIMARY KEY,
3   branch CHAR(1),
4   city VARCHAR(50),
5   customer_type VARCHAR(10),
6   gender VARCHAR(10),
7   product_line VARCHAR(50),
8   unit_price DECIMAL(10, 2),
9   quantity INT,
10  tax_5 DECIMAL(10, 4),
11  total DECIMAL(10, 4),
12  date DATE,
13  time TIME,
14  payment VARCHAR(20),
15  cogs DECIMAL(10, 2),
16  gross_margin_percentage DECIMAL(10, 9),
17  gross_income DECIMAL(10, 4),
18  rating DECIMAL(2, 1)
19 );
```

```
city VARCHAR(50),
customer_type VARCHAR(10),
gender VARCHAR(10),
product_line VARCHAR(50),
unit_price DECIMAL(10, 2),
quantity INT,
tax_5 DECIMAL(10, 4),
total DECIMAL(10, 4),
date DATE,
time TIME,
payment VARCHAR(20),
cogs DECIMAL(10, 2),
gross_margin_percentage DECIMAL(10, 9),
gross_income DECIMAL(10, 4),
rating DECIMAL(2, 1)
)
[2024-05-17 19:34:51] completed in 8 ms
```

2. Realice la carga del archivo parcial2.csv en la tabla que creó en el punto 1 dejando evidencia en pantallazos del proceso seguido (10 puntos)



✓ `select * from sales limit 5;`

invoice_id	branch	city	customer_type	gender	product_line	unit_price	quantity	tax_5	total	date
758-67-8428	A	Yangon	Member	Female	Health and beauty	74.69	7	26.1415	548.9715	2019-01
226-31-3081	C	Naypyitaw	Normal	Female	Electronic accessories	15.28	5	3.8290	88.2200	2019-03
631-41-3108	A	Yangon	Normal	Male	Home and lifestyle	46.33	7	16.2155	340.5255	2019-03
123-19-1176	A	Yangon	Member	Male	Health and beauty	58.22	8	23.2880	489.0480	2019-01
373-73-7910	A	Yangon	Normal	Male	Sports and travel	86.31	7	30.2085	634.3785	2019-02

3. Trabaje una expresión de álgebra relacional que determine la línea de producto con el producto más caro que se haya vendido (15 puntos)

Proyección de las columnas relevantes

$\pi_{\text{product_line}, \text{unit_price}}(\text{sales})$

Agregación para calcular el precio máximo por línea de producto

$\gamma_{\text{product_line}, \text{max_price} \rightarrow \text{MAX}(\text{unit_price})}(\pi_{\text{product_line}, \text{unit_price}}(\text{sales}))$

Selección del precio máximo general

$\text{max_unit_price} := \gamma_{\text{max_unit_price} \rightarrow \text{MAX}(\text{unit_price})}(\pi_{\text{unit_price}}(\text{sales}))$

Join para encontrar la línea de producto con el precio máximo

$\pi_{\text{product_line}}(\sigma_{\text{unit_price}=\text{max_unit_price}}(\text{sales} \bowtie_{\text{unit_price}=\text{max_unit_price}} \text{max_unit_price}))$

4. Construya un query en el que obtenga los totales de ventas por branch, ciudad y tipo de cliente. Tome su tiempo de ejecución (2 puntos).

```
SELECT
    branch,
    city,
    customer_type,
    SUM(total) AS total_sales
FROM
    sales
GROUP BY
    branch, city, customer_type;
```

	branch	city	customer_type	total_sales
1	A	Yangon	Normal	52562.895
2	A	Yangon	Member	53637.4755
3	C	Naypyitaw	Normal	53687.424
4	B	Mandalay	Member	53704.686
5	B	Mandalay	Normal	52492.986
6	C	Naypyitaw	Member	56881.2825

```
EXPLAIN ANALYZE
SELECT
    branch,
    city,
    customer_type,
    SUM(total) AS total_sales
FROM
    sales
GROUP BY
    branch, city, customer_type;
```

QUERY PLAN	
1	HashAggregate (cost=41.00..41.23 rows=18 width=49) (actual time=0.752..0.755 rows=6 loops=1)
2	Group Key: branch, city, customer_type
3	Batches: 1 Memory Usage: 24kB
4	-> Seq Scan on sales (cost=0.00..31.00 rows=1000 width=23) (actual time=0.007..0.131 rows=1000 loops=1)
5	Planning Time: 0.109 ms
6	Execution Time: 0.780 ms

5. Analice este query por medio de su explain (obtenga una imagen para agregarla a la resolución de su examen), y proceda a optimizar dicho query. Tome tiempo el tiempo de respuesta del query luego de la optimización (15 puntos)

```
EXPLAIN ANALYZE
SELECT
    branch,
    city,
    customer_type,
    SUM(total) AS total_sales
FROM
    sales
GROUP BY
    branch, city, customer_type;

select * from sales limit 5;
```

Output Result 149 x

6 rows

QUERY PLAN

1	HashAggregate (cost=41.00..41.23 rows=18 width=49) (actual time=0.361..0.362 rows=6 loops=1)
2	Group Key: branch, city, customer_type
3	Batches: 1 Memory Usage: 24KB
4	-> Seq Scan on sales (cost=0.00..31.00 rows=1000 width=23) (actual time=0.005..0.074 rows=1000 loops=1)
5	Planning Time: 0.085 ms
6	Execution Time: 0.382 ms

6. Construya un trigger que no permita insertar o actualizar registros en una hora que no sea “horario hábil” (de 08:00 a 20:00) (15 puntos)

May17 19:17

```
CREATE OR REPLACE FUNCTION check_business_hours()
RETURNS TRIGGER AS $$
BEGIN
    -- Verificar si la hora del registro está dentro del horario hábil (08:00 a 20:00)
    IF EXTRACT(HOUR FROM NEW.time) < 8 OR EXTRACT(HOUR FROM NEW.time) >= 20 THEN
        RAISE EXCEPTION 'Operaciones solo permitidas durante el horario hábil (08:00 a 20:00)';
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER enforce_business_hours
BEFORE INSERT OR UPDATE ON sales
FOR EACH ROW
EXECUTE FUNCTION check_business_hours();

INSERT INTO sales (invoice_id, branch, city, customer_type, gender, product_line, unit_price, quantity, tax_5, total, date, time, payment, cogs, gross_margin_percentage, gross_income, rating)
VALUES ('123-45-6789', 'A', 'Yongon', 'Normal', 'Male', 'Health and Beauty', 50.00, 1, 2.5, 52.5, '2023-05-17', '21:00:00', 'Ewallet', 50.0, 4.761904762, 2.5, 9.0);
```

[P0001] ERROR: Operaciones solo permitidas durante el horario hábil (08:00 a 20:00)
Where: PL/pgSQL function check_business_hours() line 6 at RAISE

May17 20:18

```
CREATE OR REPLACE FUNCTION check_business_hours()
RETURNS TRIGGER AS $$
BEGIN
    -- Verificar si la hora del registro está dentro del horario hábil (08:00 a 20:00)
    IF EXTRACT(HOUR FROM NEW.time) < 8 OR EXTRACT(HOUR FROM NEW.time) >= 20 THEN
        RAISE EXCEPTION 'Operaciones solo permitidas durante el horario hábil (08:00 a 20:00)';
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER enforce_business_hours
BEFORE INSERT OR UPDATE ON sales
FOR EACH ROW
EXECUTE FUNCTION check_business_hours();

INSERT INTO sales (invoice_id, branch, city, customer_type, gender, product_line, unit_price, quantity, tax_5, total, date, time, payment, cogs, gross_margin_percentage, gross_income, rating)
VALUES ('123-45-6789', 'A', 'Yongon', 'Normal', 'Male', 'Health and Beauty', 50.00, 1, 2.5, 52.5, '2023-05-17', '19:00:00', 'Ewallet', 50.0, 4.761904762, 2.5, 9.0);
```

7. Diseñe y desarrolle un reporte (vista) que muestre información de las tendencias para ser utilizado por el CEO de la empresa y la junta directiva. El diseño (columnas y filas) debe ser justificado (15 puntos)

```
82 ✓ CREATE OR REPLACE VIEW sales_trends_report AS
83 SELECT
84     branch,
85     city,
86     customer_type,
87     product_line,
88     SUM(total) AS total_sales,
89     AVG(total) AS average_sales,
90     SUM(quantity) AS total_quantity,
91     COUNT(*) AS total_transactions
92 FROM
93     sales
94 GROUP BY
95     branch, city, customer_type, product_line
96 ORDER BY
97     branch, city, customer_type, product_line;
```

sales_trends_report

Output parcial_2.public.sales_trends_report

```
.....
SUM(quantity) AS total_quantity,
COUNT(*) AS total_transactions
FROM
    sales
GROUP BY
    branch, city, customer_type, product_line
ORDER BY
    branch, city, customer_type, product_line
[2024-05-17 20:24:02] completed in 2 ms
```

Outputparcial_2.public.sales_trends_report

	branch	city	customer_type	product_line	total_sales	average_sales	total_quantity	total_transactions	
1	A	Yangon	Member	Electronic accessories	9145.689	295.0222258064516129	162		31
2	A	Yangon	Member	Fashion accessories	6942.579	315.5717727272727273	133		22
3	A	Yangon	Member	Food and beverages	8877.057	306.1054137931034483	153		29
4	A	Yangon	Member	Health and beauty	6438.894	292.677	129		22
5	A	Yangon	Member	Home and lifestyle	12556.299	369.3029117647058024	210		34
6	A	Yangon	Member	Sports and travel	9676.9575	333.6881896551724138	177		29
7	A	Yangon	Normal	Electronic accessories	9171.4245	316.2560172413793103	160		29
8	A	Yangon	Normal	Fashion accessories	9389.9295	323.7906724137931034	130		29
9	A	Yangon	Normal	Food and beverages	8286.0435	285.7256379310344828	160		29
10	A	Yangon	Normal	Health and beauty	6211.359	238.8984230769230769	129		26
11	A	Yangon	Normal	Home and lifestyle	9860.8965	318.0934354838709677	161		31
12	A	Yangon	Normal	Sports and travel	9695.742	323.1914	156		30
13	B	Mandalay	Member	Electronic accessories	7424.8545	274.9946111111111111	147		27
14	B	Mandalay	Member	Fashion accessories	7339.5315	229.360359375	143		32
15	B	Mandalay	Member	Food and beverages	9423.12	324.9351724137931034	160		29
16	B	Mandalay	Member	Health and beauty	11327.82	435.6853846153846154	172		26
17	B	Mandalay	Member	Home and lifestyle	7769.307	369.967	137		21
18	B	Mandalay	Member	Sports and travel	10420.053	347.3351	165		30
19	B	Mandalay	Normal	Electronic accessories	9626.589	343.80675	169		28
20	B	Mandalay	Normal	Fashion accessories	9073.705	302.4595	154		30
21	B	Mandalay	Normal	Food and beverages	5791.7685	275.7985	110		21
22	B	Mandalay	Normal	Health and beauty	8652.84	320.4755555555555556	148		27
23	B	Mandalay	Normal	Home and lifestyle	9779.8575	337.2364655172413793	158		29
24	B	Mandalay	Normal	Sports and travel	9568.146	299.0045625	157		32
25	C	Naypyitaw	Member	Electronic accessories	7927.9515	396.397575	120		20
26	C	Naypyitaw	Member	Fashion accessories	12041.8515	376.307859375	163		32
27	C	Naypyitaw	Member	Food and beverages	13057.443	362.70675	193		36
28	C	Naypyitaw	Member	Health and beauty	8064.3255	322.57302	127		25
29	C	Naypyitaw	Member	Home and lifestyle	7652.421	273.30075	143		28
30	C	Naypyitaw	Member	Sports and travel	8137.29	290.6175	151		28
31	C	Naypyitaw	Normal	Electronic accessories	11041.023	315.4578	213		35
32	C	Naypyitaw	Normal	Fashion accessories	9518.2185	288.4308636363636364	179		33
33	C	Naypyitaw	Normal	Food and beverages	10709.412	356.9804	176		30
34	C	Naypyitaw	Normal	Health and beauty	8551.0005	316.7037222222222222	150		27
35	C	Naypyitaw	Normal	Home and lifestyle	6243.132	367.2430588235294118	102		17

98:1 (37 chars, 3 line breaks)LFUTF-84 spaces

El diseño de la vista está diseñado para que los ejecutivos de la empresa puedan analizar cuantas ventas tienen de cada línea de productos con cada tipo de cliente en cada división o ciudad de la tienda. Por ejemplo, ya que tienen clientes miembros y clientes normales, pueden comparar en qué tiendas deben llevar más productos electrónicos o ya que tienen más ventas con los clientes normales o en qué tiendas llevar más productos de comida porque venden mucho con los clientes miembros.