HÖGSKOLAN DALARNA

Data Warehousing (AIK232)

LAB 3

REPORT

Group Members

Abu Abdullah Dhrubo

Ahmad Abdilrahim

Nahom Gebremeskel

Chinonso Williams Ubani

Introduction

In this project, we have used a dataset from a Grocery business that we used during Lab 1 for the Data Warehousing course. Furthermore, we have studied the nature and character of the data if it can fulfill all the requirements of lab-3 tasks. They identified the business process and the grains.

Aim / Tasks

The main aim of this project is to design a simple star schema for the grocery business we have chosen. To design a star schema the following four main steps were considered:

• Identifying the business process

• Identifying the grains

• Choosing the dimensions

• Choosing the measures

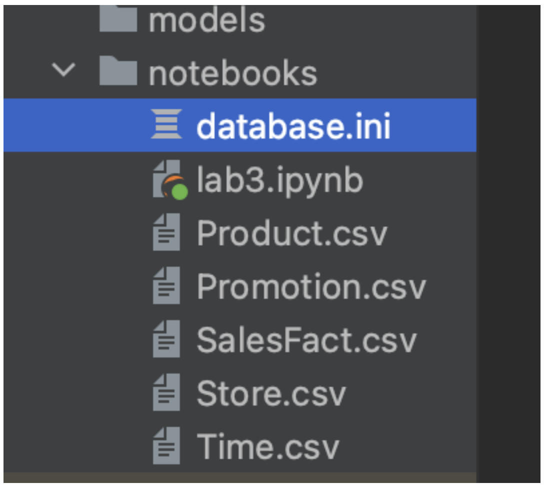
Furthermore, several tasks will be implemented at a later stage. All tasks will be presented in the implementation part of this report.

Methodology

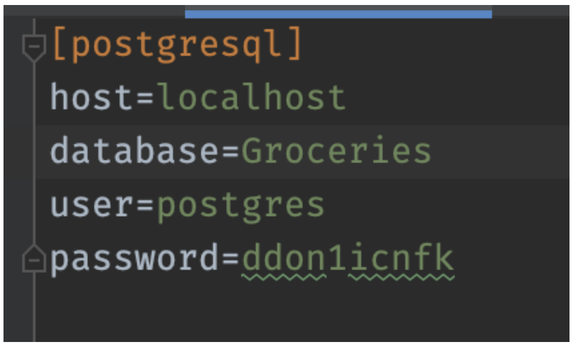
To model our database for this Grocery Store schema, we extensively utilized Python and PostgreSQL. We used python to write the database on PostgreSQL. In python, several libraries were employed depending on the usage. To mention some:

- psycopg2
- datetime
- Pandas

Setting the Environment for this project



```
models
notebooks
    database.ini
    lab3.ipynb
    Product.csv
    Promotion.csv
    SalesFact.csv
    Store.csv
    Time.csv
```



```
[postgresql]
host=localhost
database=Groceries
user=postgres
password=ddon1icnfk
```

<u>To connect to the database:</u>

1. Install PostgreSQL locally
2. Create a database called Groceries (you can rename the database anything)
3. Update the database name in the database.ini to match the database created locally
4. Update the user and password to match the local machine credentials
5. Run the Python queries/codes for the G & VG requirements

Implementation

In this section, we will provide the solution to each task and explain it in detail. To start with we designed the data model (ER Diagram) as a star schema, and we used StarUML to create it. We have defined the relationships by putting constraints on the primary key and foreign keys. Mentioned each different data type (varchar, integer, bigint, timestamp) and with these criteria, we have created the tables in PostgreSQL.

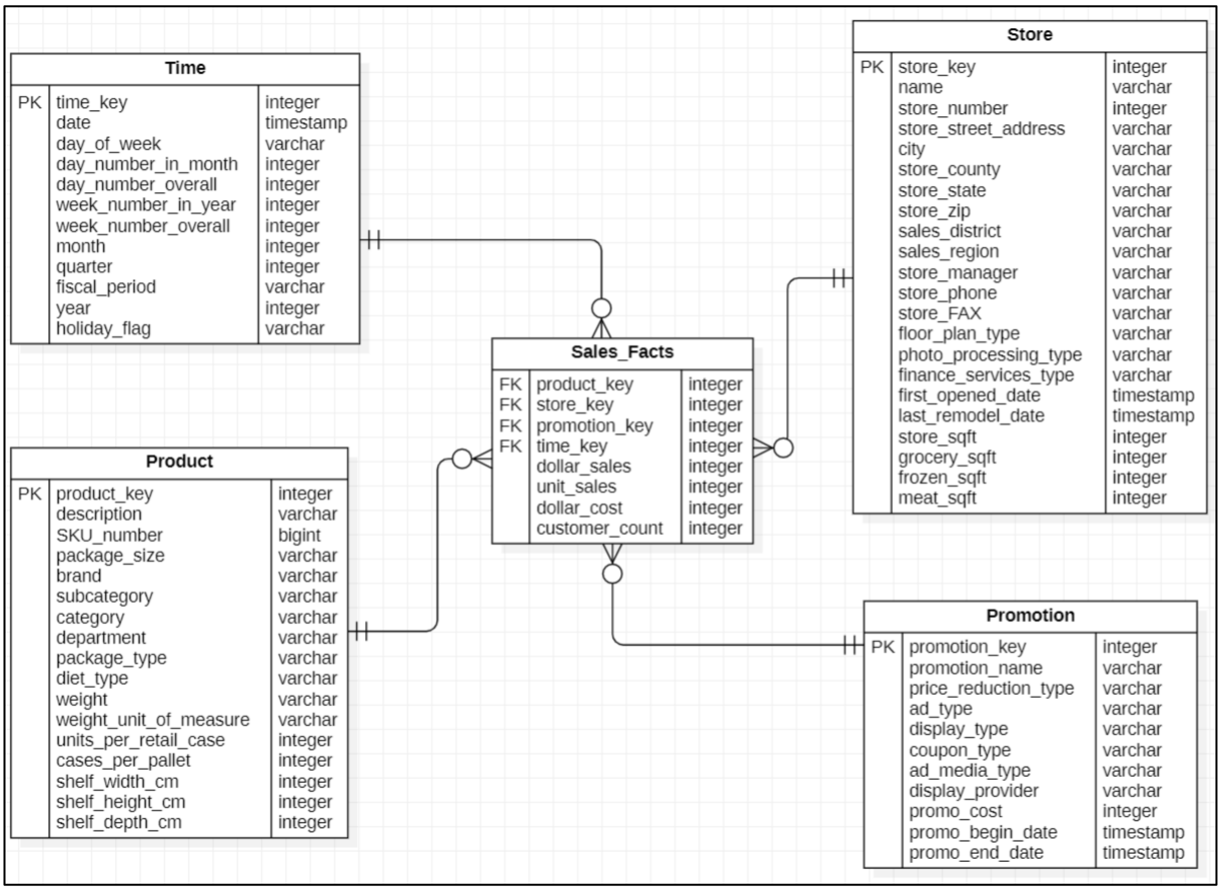


Figure. 1 ER diagram of the Grocery Business.

We have exported the Data from the Access file to separate CSVs for each entity as product, promotion, time, store, and sales facts. These files contain the product, promotion, time, store, and sales fact table for the Grocery business which we have chosen. The product file contains all the product detailed descriptions used in the grocery business; we also included the product key as the primary key for the product table. The promotion table contains all the names of promotions and types used for the business, it also includes the beginning date, end date, display type, and so on. Here the promotion key is the primary key for promotion. The time file contains the data attributes for all the dates in detail in which all products were sold in the business, this includes the day of the week, the day number in the month, the year, and so on; here the time key is used as the primary key for this table. The store file contains all the necessary details for each store location since as a grocery business it has different stores in vast areas of the country, the store is used as the primary key. Lastly, the Sales facts contain all the quantitative transactional data for the business, and all the different dimension tables have been connected as a one-to-many (1:M) relationship with the fact table.

Tasks

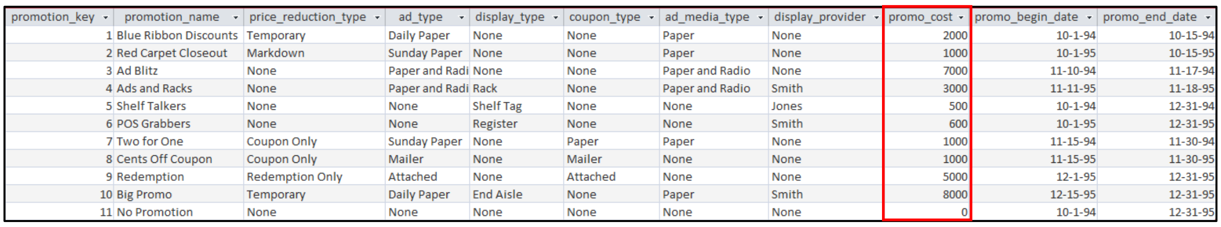
Question 0: Which transformation needs to be applied to the DW, if the company is interested in knowing the effect of promotions on sales?

To give an answer to this data the company needs to take into consideration of the following useful variables:

- Promotion name
- Year
- Quarters
- Sum of sales
- Promotion costs
- Product name as the description

So, we created a new connection between the sales and promotion tables in SQL and established the connection with the database to MS Excel. Once the connection is established, we created a pivot table in an excel file for further explorations.

We analyzed the dataset and saw that there was not a significant increase in sales across the products for this store. Also, there was not enough data to conclude the effect of promotions over sales afterward as the dataset only contain the data from Quarter 4 (October, November & December) during the years 1994 and 1995.

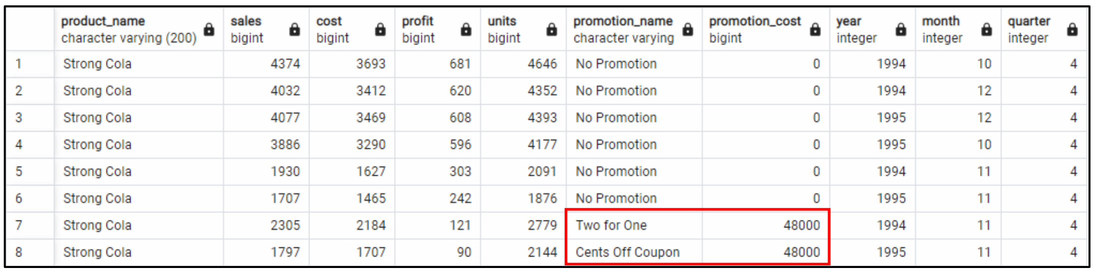


promotion_key	promotion_name	price_reduction_type	ad_type	display_type	coupon_type	ad_media_type	display_provider	promo_cost	promo_begin_date	promo_end_date
1	Blue Ribbon Discounts	Temporary	Daily Paper	None	None	Paper	None	2000	10-1-94	10-15-94
2	Red Carpet Closeout	Markdown	Sunday Paper	None	None	Paper	None	1000	10-1-95	10-15-95
3	Ad Blitz	None	Paper and Radi	None	None	Paper and Radio	None	7000	11-10-94	11-17-94
4	Ads and Racks	None	Paper and Radi	Rack	None	Paper and Radio	Smith	3000	11-11-95	11-18-95
5	Shelf Talkers	None	None	Shelf Tag	None	None	Jones	500	10-1-94	12-31-94
6	POS Grabbers	None	None	Register	None	None	Smith	600	10-1-95	12-31-95
7	Two for One	Coupon Only	Sunday Paper	None	Paper	Paper	None	1000	11-15-94	11-30-94
8	Cents Off Coupon	Coupon Only	Mailer	None	Mailer	None	None	1000	11-15-95	11-30-95
9	Redemption	Redemption Only	Attached	None	Attached	None	None	5000	12-1-95	12-31-95
10	Big Promo	Temporary	Daily Paper	End Aisle	None	Paper	Smith	8000	12-15-95	12-31-95
11	No Promotion	None	None	None	None	None	None	0	10-1-94	12-31-95

Table. 1 Promotion table

When we the below SQL query on our database we have the following output.

```sql
SELECT product.description as product_name, SUM(sales_facts.dollar_sales) as sales,
SUM(sales_facts.dollar_cost) as cost, (SUM(sales_facts.dollar_sales) - SUM(
sales_facts.dollar_cost ))as profit, SUM(sales_facts.unit_sales) as units,
COALESCE(promotion.promotion_name, 'No promotion') as promotion_name,
COALESCE(SUM(promotion.promo_cost), 0) as promotion_cost, t.year, t.month, t.quarter
FROM sales_facts
JOIN promotion ON promotion.promotion_key = sales_facts.promotion_key
JOIN product ON product.product_key = sales_facts.product_key
JOIN time t ON t.time_key = sales_facts.time_key
WHERE product.description = 'Strong Cola'
GROUP BY product_name,promotion.promotion_name,t.year, t.month, t.quarter
ORDER BY profit DESC, t.month desc, t.year desc;
```



	product_name character varying (200)	sales bigint	cost bigint	profit bigint	units bigint	promotion_name character varying	promotion_cost bigint	year integer	month integer	quarter integer
1	Strong Cola	4374	3693	681	4646	No Promotion	0	1994	10	4
2	Strong Cola	4032	3412	620	4352	No Promotion	0	1994	12	4
3	Strong Cola	4077	3469	608	4393	No Promotion	0	1995	12	4
4	Strong Cola	3886	3290	596	4177	No Promotion	0	1995	10	4
5	Strong Cola	1930	1627	303	2091	No Promotion	0	1994	11	4
6	Strong Cola	1707	1465	242	1876	No Promotion	0	1995	11	4
7	Strong Cola	2305	2184	121	2779	Two for One	48000	1994	11	4
8	Strong Cola	1797	1707	90	2144	Cents Off Coupon	48000	1995	11	4

Table. 2 The query result of aggregating by the promotion for one (strong cola) product.

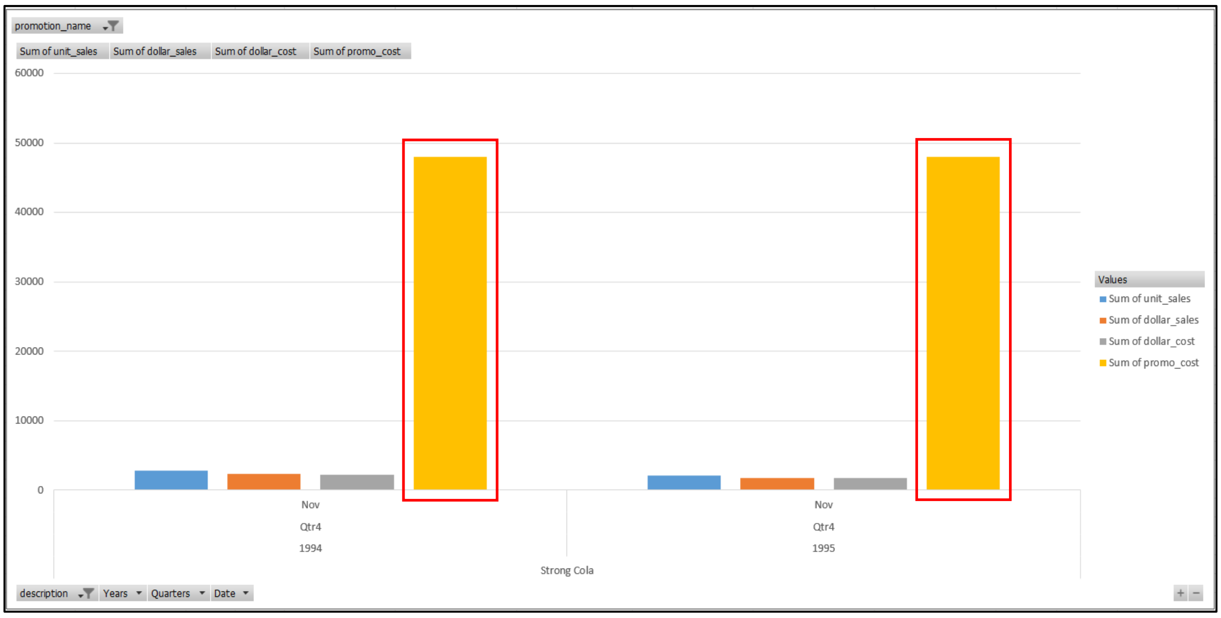
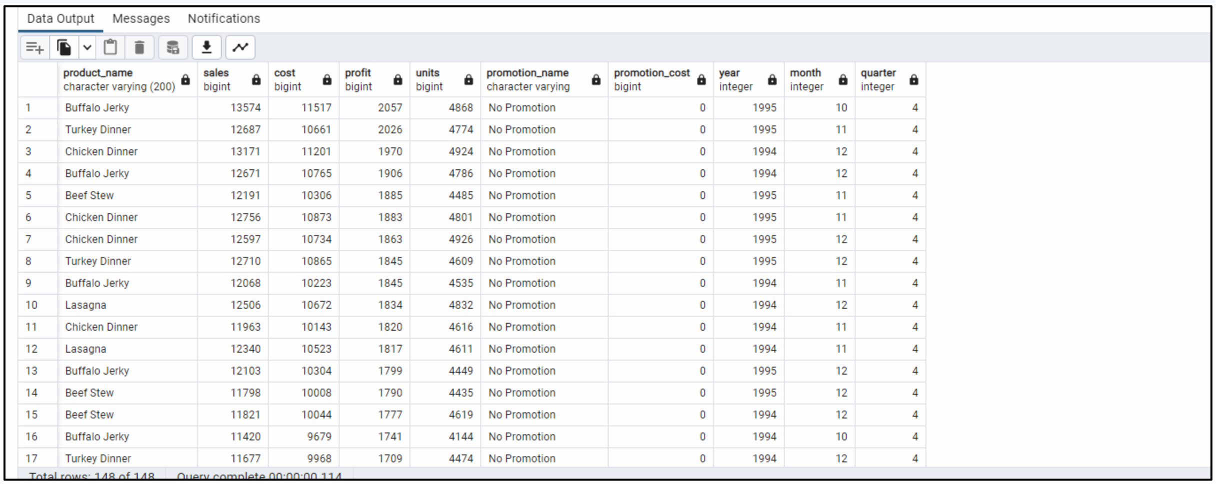


Figure 2. visualization for Strong Cola, based on the sales, cost, and promotion cost.

If we see in the highlighted section for both tables 1 & 2, we get the doubt of how one particular product has two promotions costing 48000 individually. This does not make sense as the total cost for all the promotions cost 29,100 in total to be precise, in Table. 1. And this doesn't only happen to only one product but to all of the products.

```
SELECT product.description as product_name , SUM(sales_facts.dollar_sales) as sales,
SUM(sales_facts.dollar_cost) as cost, (SUM(sales_facts.dollar_sales) - SUM(
sales_facts.dollar_cost ))as profit, SUM(sales_facts.unit_sales) as units,
COALESCE(promotion.promotion_name, 'No promotion') as promotion_name,
COALESCE(SUM(promotion.promo_cost), 0) as promotion_cost ,t.year, t.month, t.quarter
FROM sales_facts
FULL OUTER JOIN promotion ON promotion.promotion_key = sales_facts.promotion_key
FULL OUTER JOIN product ON product.product_key = sales_facts.product_key
FULL OUTER JOIN time t ON t.time_key = sales_facts.time_key
GROUP BY  product_name,promotion.promotion_name, t.year, t.month, t.quarter
ORDER BY profit DESC;
```


Data Output Messages Notifications

	product_name character varying (200)	sales bigint	cost bigint	profit bigint	units bigint	promotion_name character varying	promotion_cost bigint	year integer	month integer	quarter integer
1	Buffalo Jerky	13574	11517	2057	4868	No Promotion	0	1995	10	4
2	Turkey Dinner	12687	10661	2026	4774	No Promotion	0	1995	11	4
3	Chicken Dinner	13171	11201	1970	4924	No Promotion	0	1994	12	4
4	Buffalo Jerky	12671	10765	1906	4786	No Promotion	0	1994	12	4
5	Beef Stew	12191	10306	1885	4485	No Promotion	0	1995	11	4
6	Chicken Dinner	12756	10873	1883	4801	No Promotion	0	1995	11	4
7	Chicken Dinner	12597	10734	1863	4926	No Promotion	0	1995	12	4
8	Turkey Dinner	12710	10865	1845	4609	No Promotion	0	1995	12	4
9	Buffalo Jerky	12068	10223	1845	4535	No Promotion	0	1994	11	4
10	Lasagna	12506	10672	1834	4832	No Promotion	0	1994	12	4
11	Chicken Dinner	11963	10143	1820	4616	No Promotion	0	1994	11	4
12	Lasagna	12340	10523	1817	4611	No Promotion	0	1994	11	4
13	Buffalo Jerky	12103	10304	1799	4449	No Promotion	0	1995	12	4
14	Beef Stew	11798	10008	1790	4435	No Promotion	0	1995	12	4
15	Beef Stew	11821	10044	1777	4619	No Promotion	0	1994	12	4
16	Buffalo Jerky	11420	9679	1741	4144	No Promotion	0	1994	10	4
17	Turkey Dinner	11677	9968	1709	4474	No Promotion	0	1994	12	4

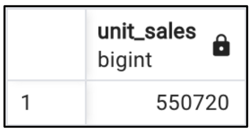
Total rows: 148 of 148 Query complete 00:00:00.114

Table. 3 aggregated values for the products with no promotion.

So, depending on these above findings, we can conclude that the way of storing the values of promotion cost has caused such misinformation. In order to fix this the database needs a transformation by changing the method for storing the promotion cost differently. One possible way could be to split the promotion cost per unit so that when we aggregate the distribution of promotion cost, the sum would be 29100 instead of some wrong calculation. And this will help the company analyze the effect of promotion on sales.

Question 1: How many sales did the company have?

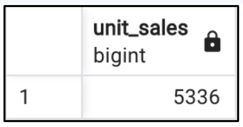
```sql
SELECT SUM(unit_sales) AS unit_sales
FROM sales_facts;
```



	unit_sales bigint
1	550720

Question 2: How many products did we sell in a store last month?

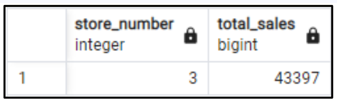
```sql
SELECT SUM(unit_sales) AS unit_sales
FROM sales_facts
WHERE time_key in (SELECT time_key FROM time WHERE date BETWEEN '1995-12-01' AND '1995-12-31')
AND store_key in (SELECT store_key FROM store WHERE store_number = '10');
```



	unit_sales bigint
1	5336

Question 3: Which one of our stores has the highest amount of sales?

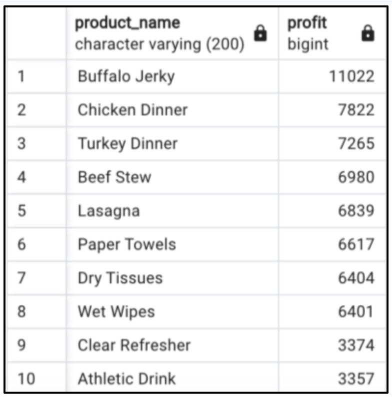
```sql
SELECT store_number, total_sales
FROM (
  SELECT s.store_number, SUM(sf.dollar_sales) as total_sales
  FROM sales_facts sf
  JOIN store s ON s.store_key = sf.store_key
  GROUP BY s.store_key
  ORDER BY total_sales DESC
  LIMIT 1
) t;
```



	store_number integer	total_sales bigint
1	3	43397

Question 4: Which products are the most lucrative?

```sql
SELECT product.description as product_name , SUM(sales_facts.dollar_sales) -
SUM(sales_facts.dollar_cost) as profit
FROM sales_facts
JOIN product ON product.product_key = sales_facts.product_key
GROUP BY product.description
ORDER BY profit desc;
```

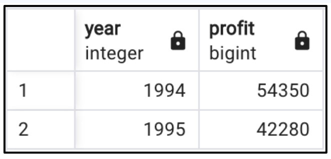


	product_name character varying (200)	profit bigint
1	Buffalo Jerky	11022
2	Chicken Dinner	7822
3	Turkey Dinner	7265
4	Beef Stew	6980
5	Lasagna	6839
6	Paper Towels	6617
7	Dry Tissues	6404
8	Wet Wipes	6401
9	Clear Refresher	3374
10	Athletic Drink	3357

Question 5: *What was the most lucrative day, month, or year?*

Year:

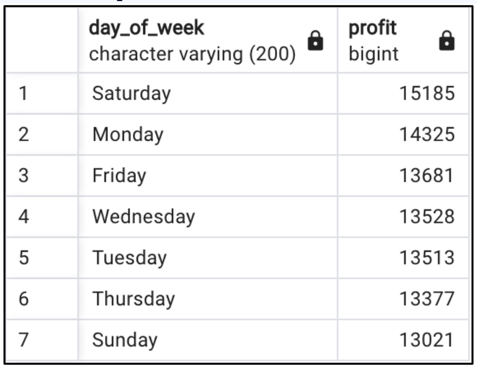
```
SELECT time.year, SUM(sales_facts.dollar_sales) - SUM(sales_facts.dollar_cost) as profit
FROM sales_facts
JOIN time ON time.time_key = sales_facts.time_key
GROUP BY time.year
ORDER BY profit desc;
```



	year integer	profit bigint
1	1994	54350
2	1995	42280

Day of the week:

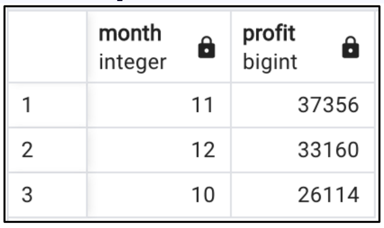
```
SELECT time.day_of_week, SUM(sales_facts.dollar_sales) - SUM(sales_facts.dollar_cost) as
profit
FROM sales_facts
JOIN time ON time.time_key = sales_facts.time_key
GROUP BY time.day_of_week
ORDER BY profit desc;
```



	day_of_week character varying (200)	profit bigint
1	Saturday	15185
2	Monday	14325
3	Friday	13681
4	Wednesday	13528
5	Tuesday	13513
6	Thursday	13377
7	Sunday	13021

Month:

```
SELECT time.month, SUM(sales_facts.dollar_sales) - SUM(sales_facts.dollar_cost) as profit
FROM sales_facts
JOIN time ON time.time_key = sales_facts.time_key
GROUP BY time.month
ORDER BY profit desc;
```



	month integer	profit bigint
1	11	37356
2	12	33160
3	10	26114