

# Additional Feature Works:

## Part 1:

### Web Scraping and Data Storage Application:

The Python script facilitated web scraping from Best Buy, depicted in the provided screenshot. It extracted product data from the website and stored it in a local database table named "best\_buy\_products". This automated process streamlined data acquisition from Best Buy's platform, enhancing efficiency in gathering product information. The script's functionality enabled seamless integration of scraped data into the database, facilitating further analysis or utilization as needed. Overall, the implementation showcased effective utilization of Python for web scraping and database management, exemplifying the script's practical application in data retrieval tasks.

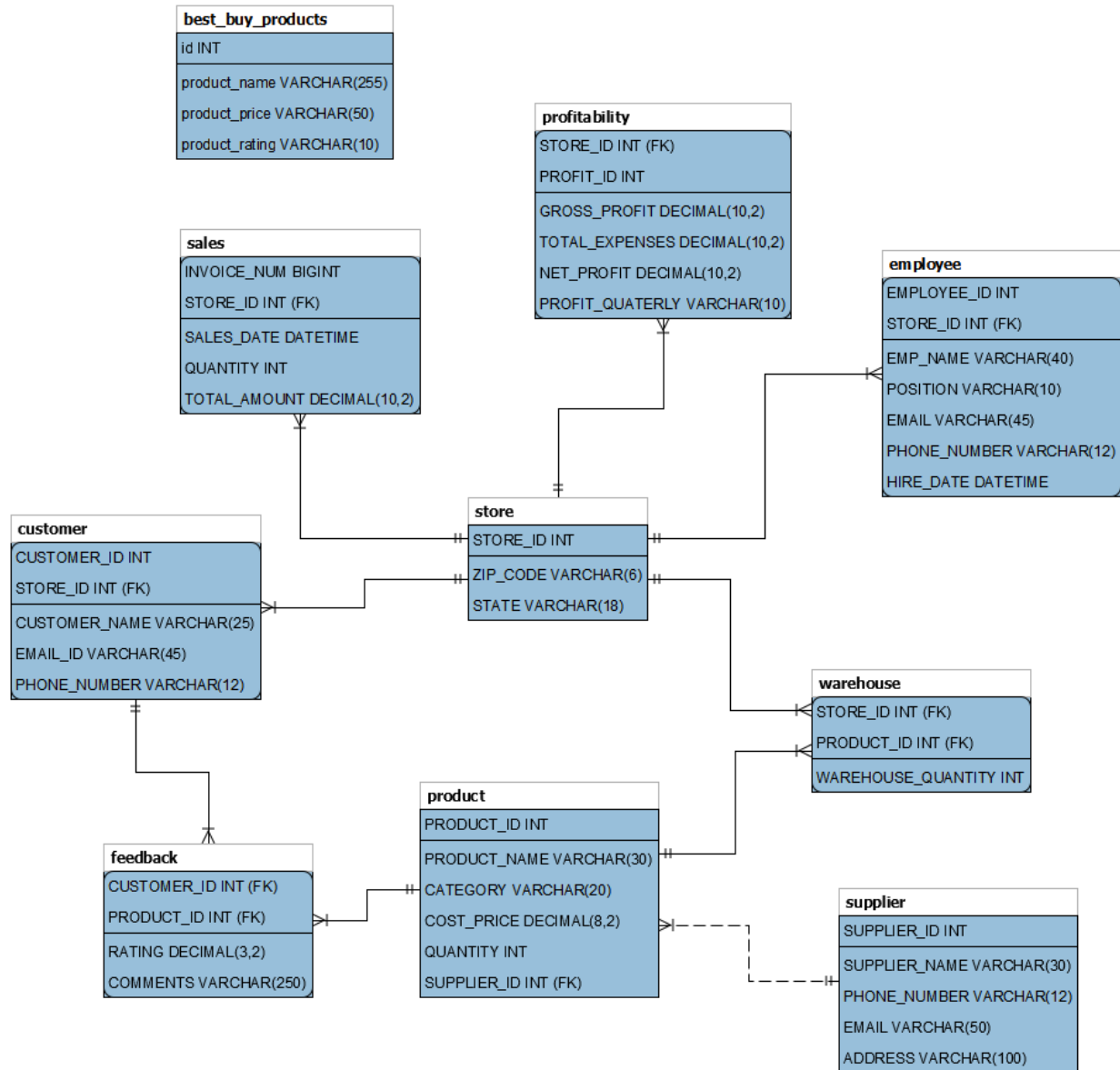
The screenshot displays a MySQL IDE interface with the following components:

- Navigator:** Shows the database structure for 'mm\_team04\_02', including tables like 'best\_buy\_products', 'customer', 'employee', 'feedback', and 'product'.
- SQL Editor:** Contains a query file named 'Milestone2Queries' with the following SQL code:

```
1 # Checking Data
2 • USE mm_team04_02;
3 • SELECT * from best_buy_products;
```
- Result Grid:** Displays the results of the query, showing a table with columns 'id', 'product\_name', 'product\_price', and 'product\_rating'. The results include data for various iPhone models and their prices and ratings.
- Output:** Shows the execution output, including the table name 'best\_buy\_products' and the results of the query.

id	product_name	product_price	product_rating
76	Apple - iPhone 15 Pro Max 1TB - Black Titanium ...	\$44.45	4.7
77	Apple - iPhone 15 Pro Max 1TB - Natural Titanium...	\$44.44	4.4
78	Apple - iPhone 14 Plus 128GB - Midnight (Verizon)	\$23.05	4.9
79	Apple - iPhone 14 128GB - Blue (AT&T)	\$20.28	4.7
80	Apple - iPhone 14 Plus 128GB - Midnight (AT&T)	\$23.06	4.7

Here is the Model of our database where you can observe that the table from Best Buy has been integrated into the database.



## Part 2:

### Moving Data from Relational Database to MongoDB:

Through Python code, we successfully transferred data from a Relational Database to MongoDB, as evidenced by the provided screenshot. The migration encompassed records from three tables—Customer, Sales, and Store—contained within the EliteWork database. This seamless data transfer demonstrates the versatility of Python in bridging dissimilar database systems, enabling smooth transition and integration of relational data into MongoDB's document-oriented structure. The screenshot illustrates the presence of two tables within the EliteWork database post-transfer, underscoring the effectiveness of the Python script in facilitating cross-database data movement, a critical aspect of modern data management workflows.

The screenshot displays the MongoDB Compass interface for a local connection at localhost:27017. The left sidebar shows the database structure, with the 'EliteWork' database selected. Under 'EliteWork', three collections are listed: 'Customer', 'Sales', and 'Store'. The main panel shows the details for these collections, including storage size, document count, average document size, index count, and total index size.

Collection	Storage size	Documents	Avg. document size	Indexes	Total index size
Customer	77.82 kB	1 K	117.00 B	1	24.58 kB
Sales	45.06 kB	1 K	105.00 B	1	32.77 kB
Store	20.48 kB	20	48.00 B	1	20.48 kB

## Part 3:

### Moving Data from CSSQL Database to Amazon RDS Database

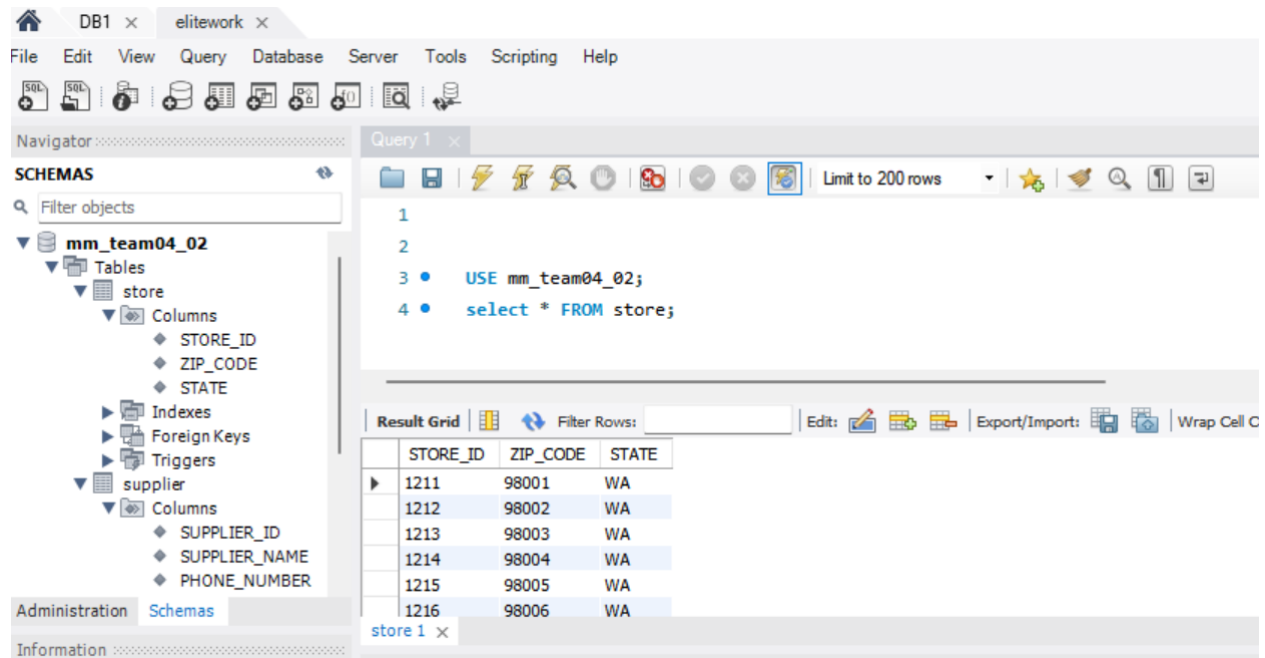
Utilizing a Python script, we established a connection from CSSQL to an Amazon RDS Database, facilitating the seamless transfer of two tables—Store and Suppliers—from the mm\_team04\_02 database. The process was executed under the database connection name "elitework".

The AWS RDS Database credentials provided were as follows:

Username: admin

Password: Bhusal9891t

This successful data migration exemplifies the script's efficacy in bridging disparate database environments and underscores its utility in facilitating smooth data transfers between CSSQL and Amazon RDS Database, thereby enhancing data management capabilities.



The screenshot displays a database management interface with a menu bar (File, Edit, View, Query, Database, Server, Tools, Scripting, Help) and a toolbar. The left sidebar shows a tree view of the database structure for 'mm\_team04\_02', including tables 'store' and 'supplier'. The main area shows a query window with the following SQL code:

```
1  
2  
3 • USE mm_team04_02;  
4 • select * FROM store;
```

Below the query window is a 'Result Grid' showing the data from the 'store' table. The grid has columns for STORE\_ID, ZIP\_CODE, and STATE. The data is as follows:

STORE_ID	ZIP_CODE	STATE
1211	98001	WA
1212	98002	WA
1213	98003	WA
1214	98004	WA
1215	98005	WA
1216	98006	WA

The bottom of the interface shows tabs for 'Administration', 'Schemas', and 'Information'.