**Report**

**Introduction:**

This database management system was created for a coffee shop business. Such entities as order, order\_details, barista, server, customer, items, bill, coffee\_shop, and manager were included in the system:

1. Order: An order represents a customer's request for items from the coffee shop. It contains information such as the order\_id, barista\_id, customer\_id, server\_id, item\_id. The primary key is order\_id and all other attributes are foreign keys.

Orders

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| order\_id | server\_id | customer\_id | barista\_id | Item\_id | completed | received | total\_bill |
| 1 | 3 | 1 | 3 | 1 | True | True | 10 |
| 2 | 2 | 2 | 3 | 2 | True | False | 20 |
| 3 | 1 | 1 | 2 | 1 | False | false | 30 |

1. It’s in 1NF, because all columns are in atomic form
2. It’s doesn’t satisfy 2NF, because attributes completed, received and total\_bill don’t depend on entire primary key. They only depend on order\_id attribute. So, we divide it into to tables.

Orders

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| order\_id | server\_id | customer\_id | barista\_id | item\_id |
| 1 | 3 | 1 | 3 | 1 |
| 2 | 2 | 2 | 3 | 2 |
| 3 | 1 | 1 | 2 | 1 |

Order\_Details

|  |  |  |  |
| --- | --- | --- | --- |
| order\_id | completed | received | total\_bill |
| 1 | True | True | 10 |
| 2 | True | False | 20 |
| 3 | False | false | 30 |

1. It’s in 3NF, because there is no transitive dependency among non-key attributes.

2. Order\_details: The order\_details entity contains detailed information about items ordered by customer. It includes such details as order\_id, completed, received, total\_bill. Primary key is order\_id.

3. Barista: A barista is a person responsible for preparing and serving coffee to customers. The barista entity contains information such as the barista ID, name, and contact information.

Barista

|  |  |  |  |
| --- | --- | --- | --- |
| barista\_id | name | phone\_num | address |
| 1 | Sherkhan | 8771 562 12 45 | Saina 6 |
| 2 | Aman | 8 705 958 12 78 | Tole bi 56 |
| 3 | Ernar | 8747 569 32 45 | Alfarabi 87 |

Barista table in

1. 1NF because all columns are in atomic form
2. 2NF because all attributes depend on entire key
3. 3NF because there is no transitive dependency among non-key attributes.
4. Server: A server is a person responsible for taking orders and serving customers. The server entity contains information such as the server ID, name, and contact information.

Server

|  |  |  |  |
| --- | --- | --- | --- |
| server\_id | name | phone\_num | address |
| 1 | Nurdaulet | 8 771 562 32 45 | Saina 6 |
| 2 | Dias | 8 705 958 65 78 | Tole bi 56 |
| 3 | Abylai | 8 747 569 87 45 | Alfarabi 87 |

Server table in

1. 1NF because all columns are in atomic form
2. 2NF because all attributes depend on entire key
3. 3NF because there is no transitive dependency among non-key attributes.

5. Customer: A customer is a person who visits the coffee shop and makes an order. The customer entity contains information such as the customer ID, name, and contact information.

Customers

|  |  |
| --- | --- |
| customer\_id | name |
| 1 | Asset |
| 2 | Ansar |

Customers table in

1. 1NF because all columns are in atomic form
2. 2NF because all attributes depend on entire key
3. 3NF because there is no transitive dependency among non-key attributes.

6. Items: Items represent the products sold at the coffee shop, such as coffee, tea, sandwiches etc. The items entity contains information such as the item ID, name, description, and cost and quantity.

Items

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| item\_id | name | description | cost | quantity |
| 1 | Americano | Tasty hot | 10 | 80 |
| 2 | Latte | Creamy hot | 15 | 90 |
| 3 | Morocco tea | uf kefteme | 11 | 210 |

Items table in

1. 1NF because all columns are in atomic form
2. 2NF because all attributes depend on entire key
3. 3NF because there is no transitive dependency among non-key attributes.

7. Bill: A bill represents the total amount due for an order. The bill entity contains information such as the bill ID, the order ID, the date and price of the bill.

Bill

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| bill\_id | order\_id | description | date | Price |
| 1 | 1 | Americano | 17.04.2023 | 10 |
| 2 | 2 | Latte | 18.04.2023 | 20 |

Bill table in

1. 1NF because all columns are in atomic form
2. 2NF because all attributes depend on entire key
3. 3NF because there is no transitive dependency among non-key attributes.

8. Coffee\_shop: The coffee\_shop entity contains information about the coffee shop, such as its name, address, and contact information.

Coffee\_shop

|  |  |  |  |
| --- | --- | --- | --- |
| shop\_id | name | phone\_num | address |
| 1 | Tea Dot | 8 771 741 85 45 | Saina 6 |
| 2 | Coffee Boom | 8 705 958 61 28 | Tole bi 56 |
| 3 | Wake cup | 8 747 569 87 95 | Alfarabi 87 |

Coffee\_shop table in

1. 1NF because all columns are in atomic form
2. 2NF because all attributes depend on entire key
3. 3NF because there is no transitive dependency among non-key attributes.

9. Manager: A manager is a person responsible for managing the coffee shop. The manager entity contains information such as the manager ID, name, address, and contact information.

Manager

|  |  |  |  |
| --- | --- | --- | --- |
| manager\_id | name | phone\_num | address |
| 1 | Taukekhan | 8771 484 32 45 | Saina 6 |
| 2 | Yerbol | 8 705 122 65 78 | Tole bi 56 |
| 3 | Bissenbay | 8747 214 87 45 | Alfarabi 87 |

Managers table in

1. 1NF because all columns are in atomic form
2. 2NF because all attributes depend on entire key
3. 3NF because there is no transitive dependency among non-key attributes.

Database Design: The database design for the coffee shop DBMS includes the following tables:

1. Order table: This table contains the following columns:

Order ID (primary key)

Customer ID (foreign key)

Server ID (foreign key)

Barista ID (foreign key)

Item ID (foreign key)

2. Order\_details table: This table contains the following columns:

Order detail ID (primary key)

Completed

Receiver

Total\_bill

3. Barista table: This table contains the following columns:

Barista ID (primary key)

Name

Phone

Address

4. Server table: This table contains the following columns:

Server ID (primary key)

Name

Phone

Address

5. Customer table: This table contains the following columns:

Customer ID (primary key)

Name

6. Items table: This table contains the following columns:

Item ID (primary key)

Name

Description

Cost

Quantity

7. Bill table: This table contains the following columns:

Bill ID (primary key)

Order ID (foreign key)

Date

Time

Price

8. Coffee\_shop table: This table contains the following columns:

Coffee shop ID (primary key)

Name

Phone

Address

9. Manager table: This table contains the following columns:

Manager ID (primary key)

Name

Phone

Address

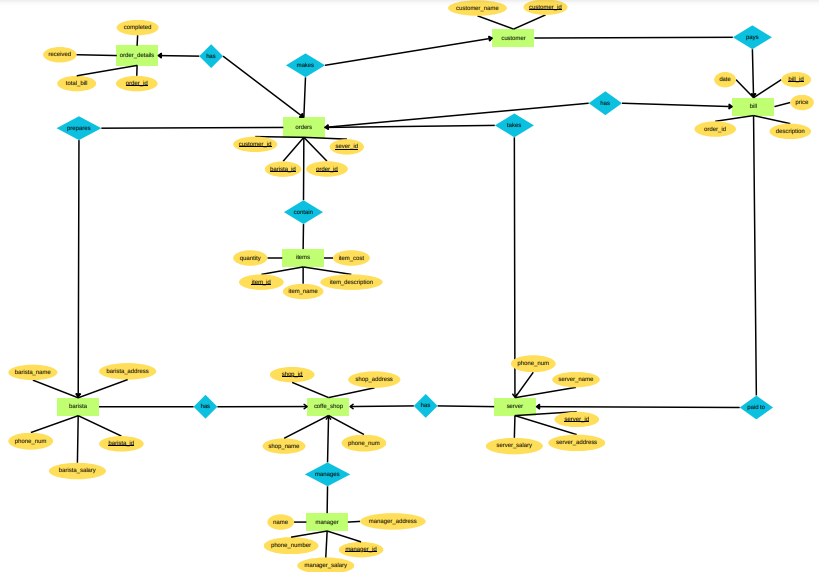
Data Flow: The coffee shop DBMS operates as follows:

1. A customer places an order by providing their customer ID, item IDs, and quantities.

2. The order details are stored in the order\_details table.

3. The order ID, customer ID, and date and time are stored in the order table.

Please refer to the database schema for a more detailed description of the system's design



1.Customer has attributes customer\_name and customer\_id. Primary key of this table is customer\_id. Relation between table Orders is one-to-many. One customer can make multiple orders, but one order made by only one customer. Relation between tables Customers and bill is also one-to-one.

2.Orders has attributes order\_id, server\_id, barista\_id, customer\_id. Primary key of table Orders is order\_id. Orders table has the attribute server\_id, which is a foreign key referencing the id attribute in the Server table. Relation between table Orders and Server is one-to-many. Orders table has the attr ibute barista\_id, which is a foreign key referencing the id attribute in the Barista table. Relation between tables Orders and Barista is also one-to-many. Orders table has the attribute customer\_id, which is a foreign key referencing the id attribute in the Customer table. Relation between tables Orders and Customer is one-to-many too. Orders contain Items, relation between these tables is many-to-many. Table Orders has Bill, relation between them is one-to-one. Orders has Order\_details, and relation between Orders and Order\_details is one-to-one.

3.Order\_details has attributes total\_bill, completed, received and order\_id, which is foreign key referencing the id attribute in the Orders. Relation between Order\_details and Orders is one-to-one.

4.Items has attributes item\_cost, quantity, item\_name, item\_description and item\_id, which is primary key of Items table. Oredrs contain Items. Relation between tables Orders and Items is many-to-many.

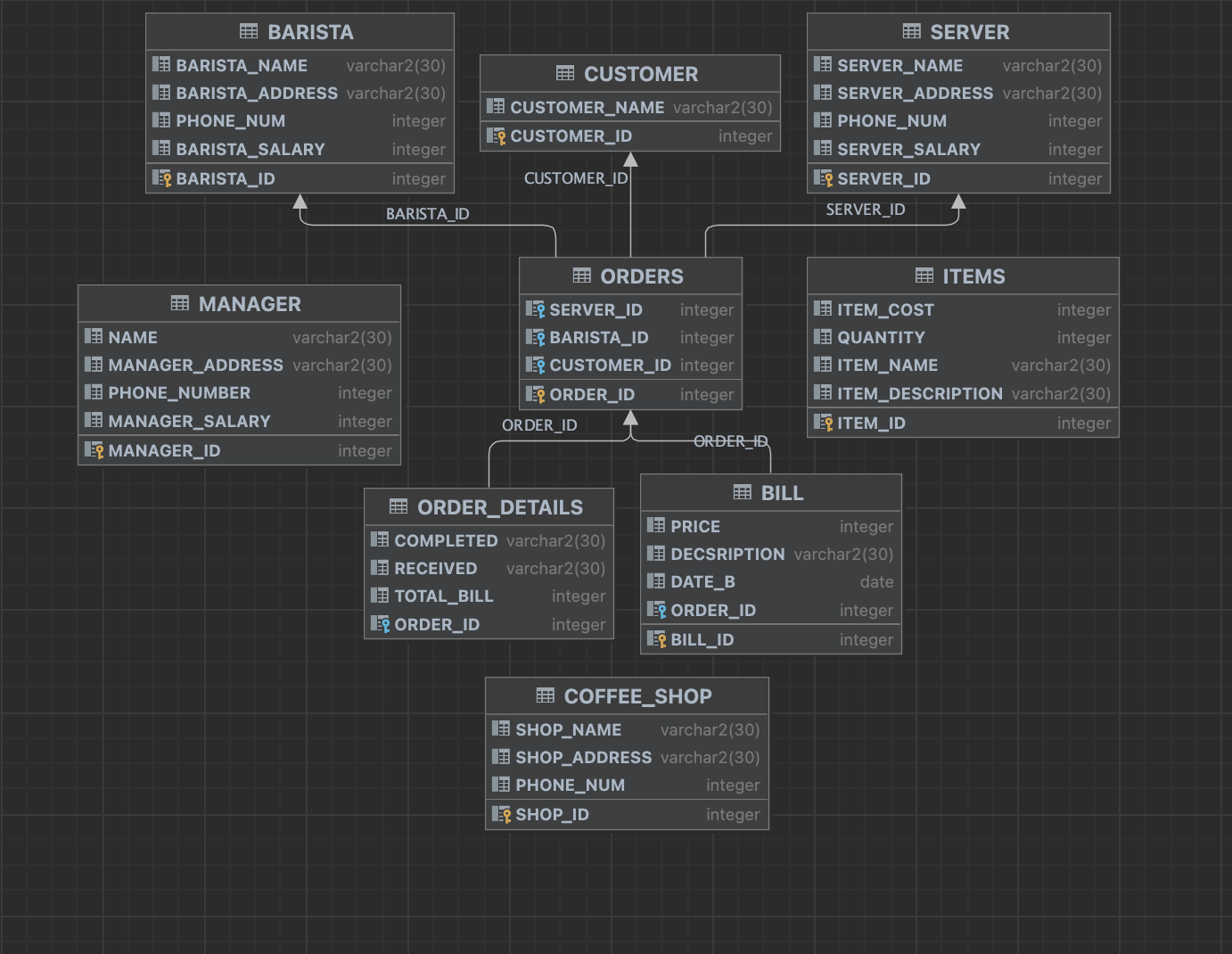
5.Bill has attributes price, description, date, bill\_id, which is primary key of table bill and order\_id, which is foreign key referencing the id attribute in the Order table. Bill is paid to Server. Relation between Bill and Server is one-to-many. Bill is paid by Customer. Relation between Bill and Customer is one-to-many.

6.Barista has attributes barista\_name, phone\_num, barista\_salary, barista\_address, barista\_salary and barista\_id, which is primary key of Barista table. Barista works at Coffee\_shop. Relation between Coffee\_shop and Barista is one-to-many. Barista prepares an Orders. Relation between table Barista and Orders is one-to-many.

7.Coffee\_shop has shop\_address, phone\_num, shop\_name and shop\_id, which is primary key of table. Coffee\_shop has Barista, Server and Manager. Relations between Coffee\_shop and Server, Coffee\_shop and Barista and Coffee\_shop and Manager are one-to-many.

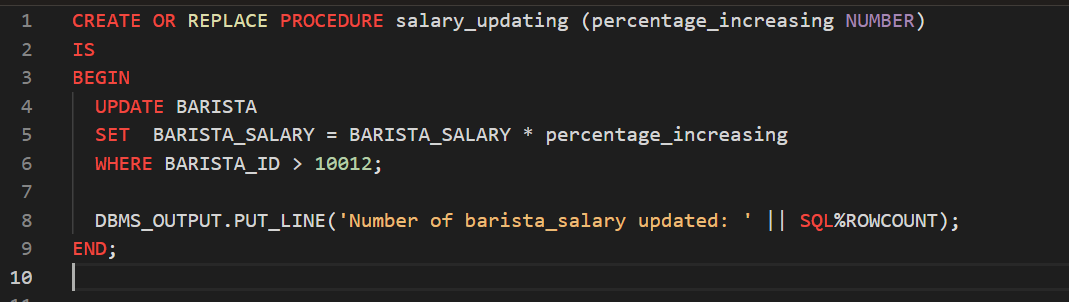
8.Server has attributes server\_name, phone\_num, server\_salary, server\_address, server\_salary and server\_id, which is primary key of Server table. Server works at Coffee\_shop. Relation between Coffee\_shop and Server is one-to-many. Server takes an Orders. Relation between table Server and Orders is one-to-many.

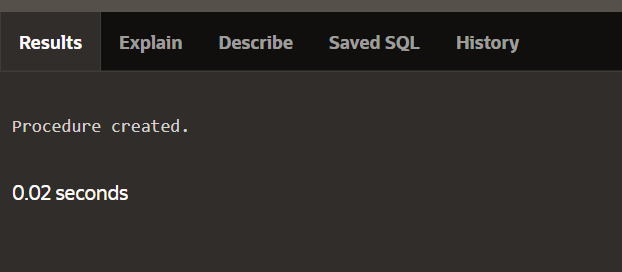
9.Manager has attributes phone\_num, manager\_salary, name, manager\_address, manager\_salary and manager\_id, which is primary key of Manager table. Manager manages Coffee\_shop. Relation between Coffee\_shop and Manager is one-to-many.



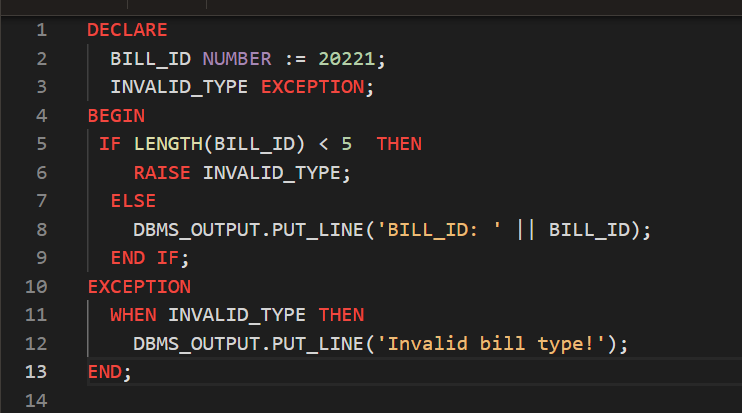
**3.Coding Part**

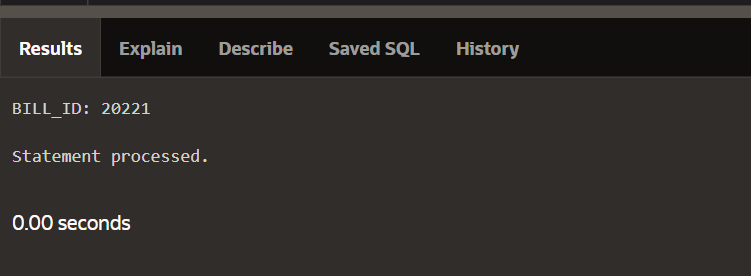
1) Procedure which uses SQL%ROWCOUNT to determine the number of rows affected

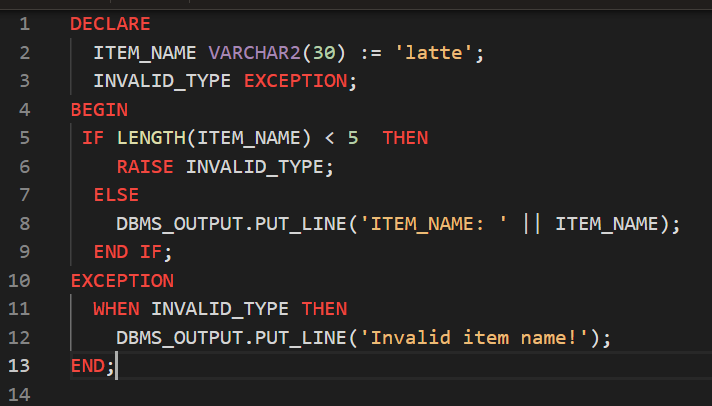


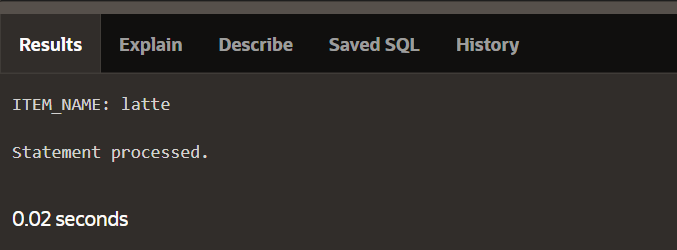


2) Add user-defined exception which disallows to enter title of item (e.g. book) to be less than 5 characters

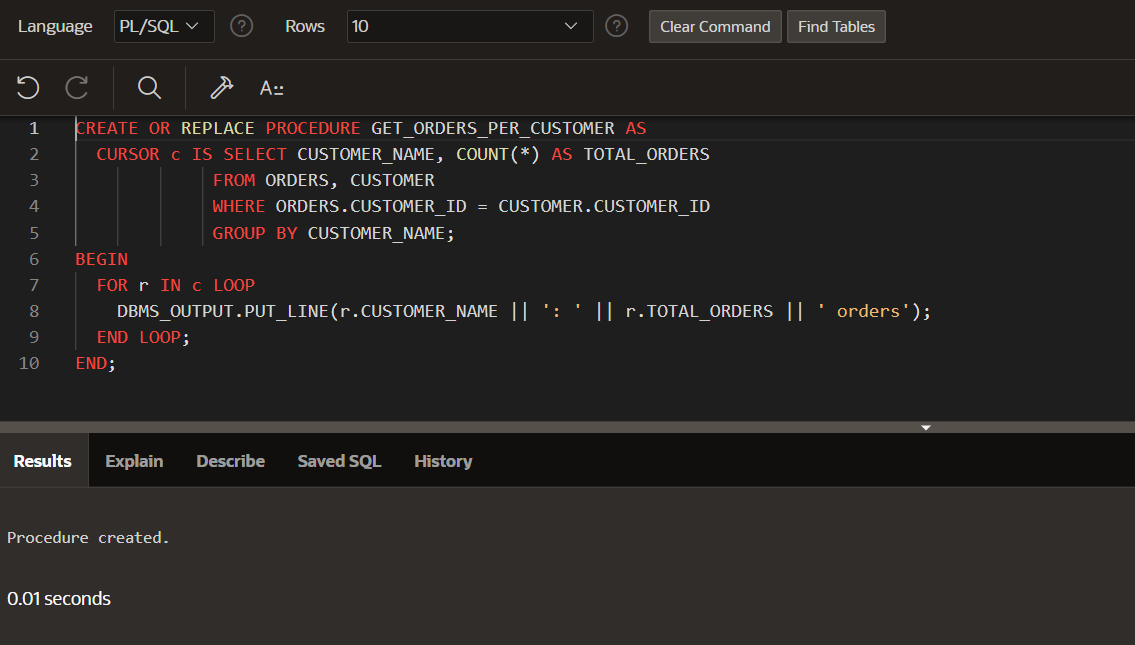


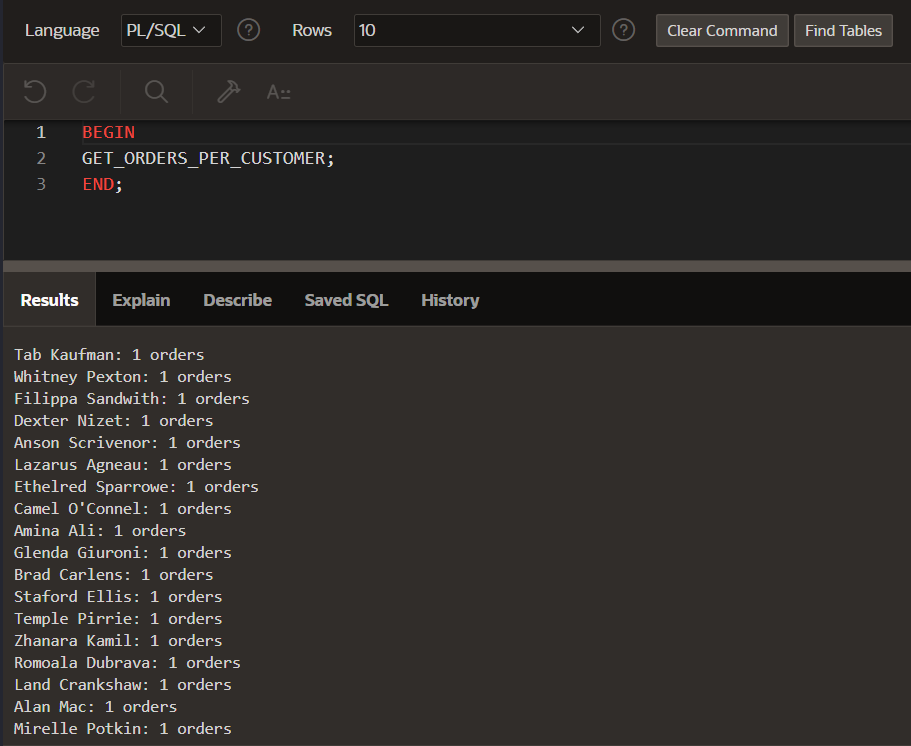






3) Procedure which uses GROUP BY





This procedure is designed to retrieve the number of orders per customer from the **orders** and **customer** tables. The output of the procedure will be a list of customer names and the number of orders they have placed.

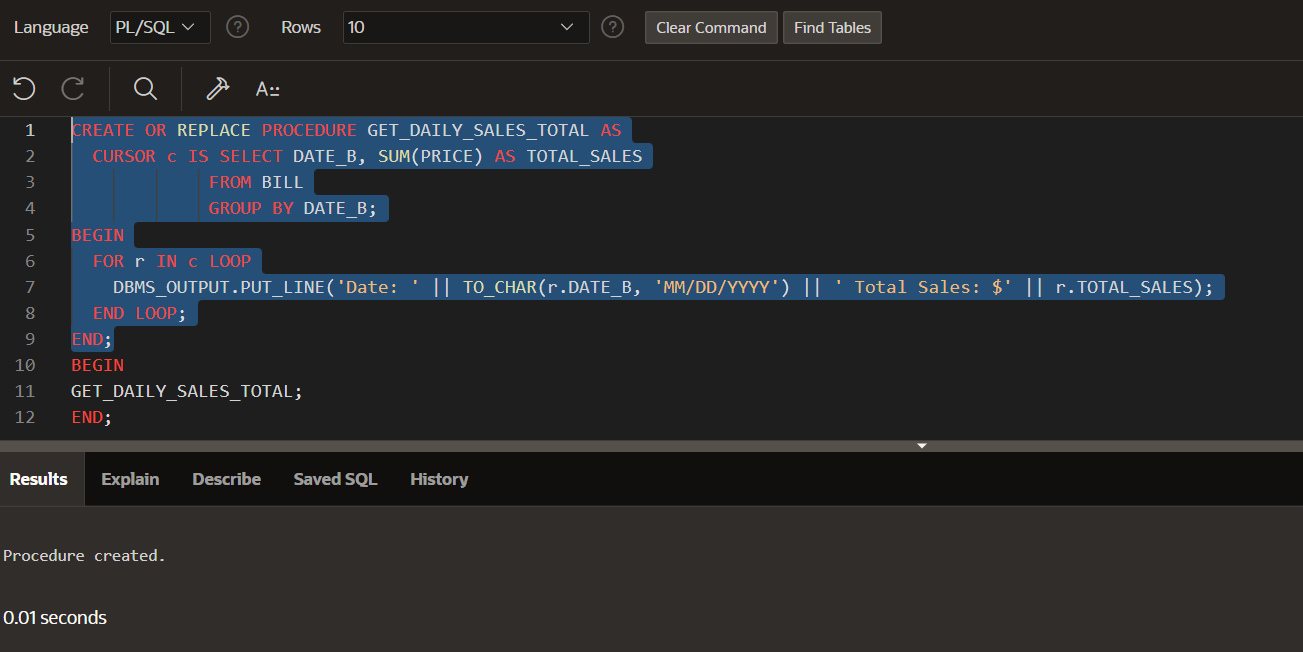
**Why do we need this procedure?**

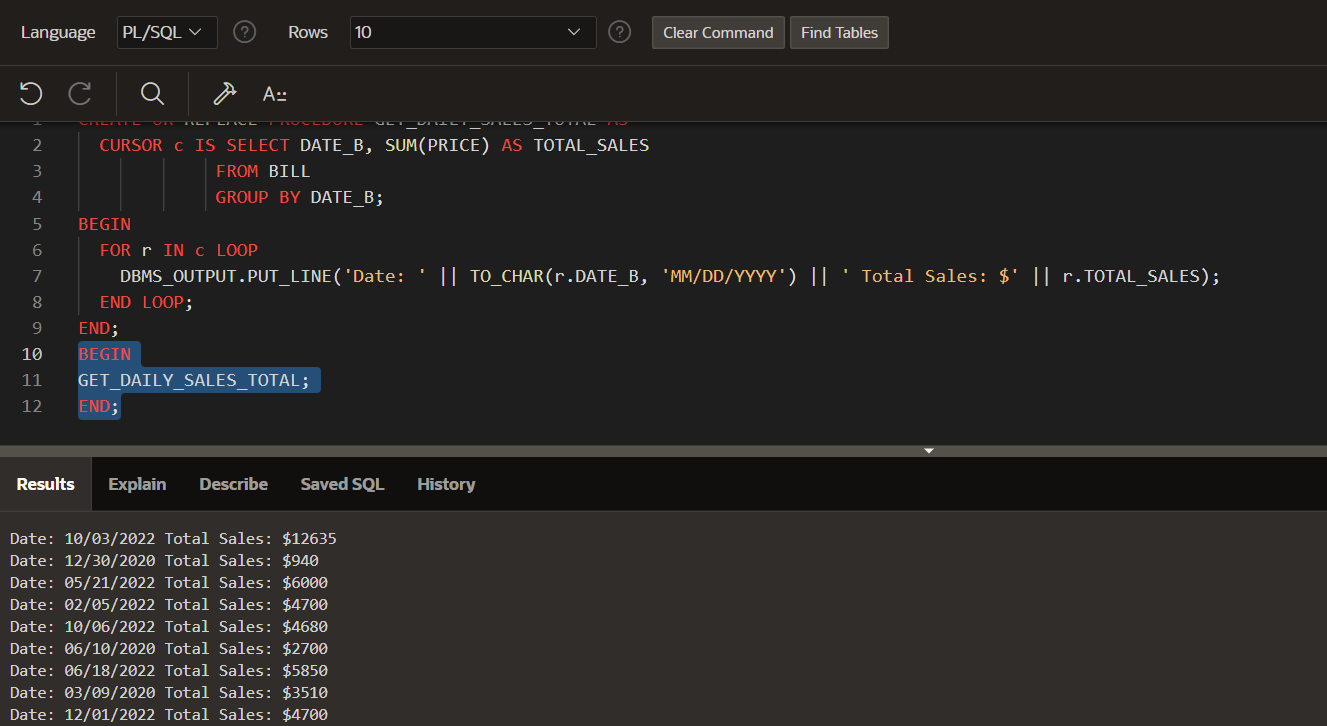
This procedure is useful for analyzing the customer order data for a coffee shop. By retrieving the number of orders for each customer, you can identify your most loyal customers and tailor your marketing and loyalty programs to their needs.

Explanation:  
First, we declare a cursor named **c\_order\_summary**. This cursor retrieves data from the **customer, orders**, and **bill** tables. Specifically, it selects the **customer\_name**, the count of orders for each customer (**total\_orders**).

Inside the **begin** block, we use a loop to iterate through each row returned by the **c\_order\_summary** cursor.

Inside the loop, we use the **dbms\_output.put\_line** procedure to display the results for each customer. We display the customer name, total orders.





This procedure is designed to retrieve the total sales for each day from the **BILL** table. The output of the procedure will be a list of dates and their corresponding total sales.

**Why do we need this procedure?**

This procedure is useful for analyzing the daily sales data for a coffee shop. By retrieving the total sales for each day, you can identify which days are the busiest and adjust staffing and inventory accordingly.

Explanation:

Firstly, we create new procedure named **get\_daily\_sales\_total**

The **cursor** statement declares a cursor named **daily\_sales\_cursor** that retrieves data from the **bill** table. The cursor selects the **date\_b** column and the **sum** of the **price** column, grouped by **date\_b.**

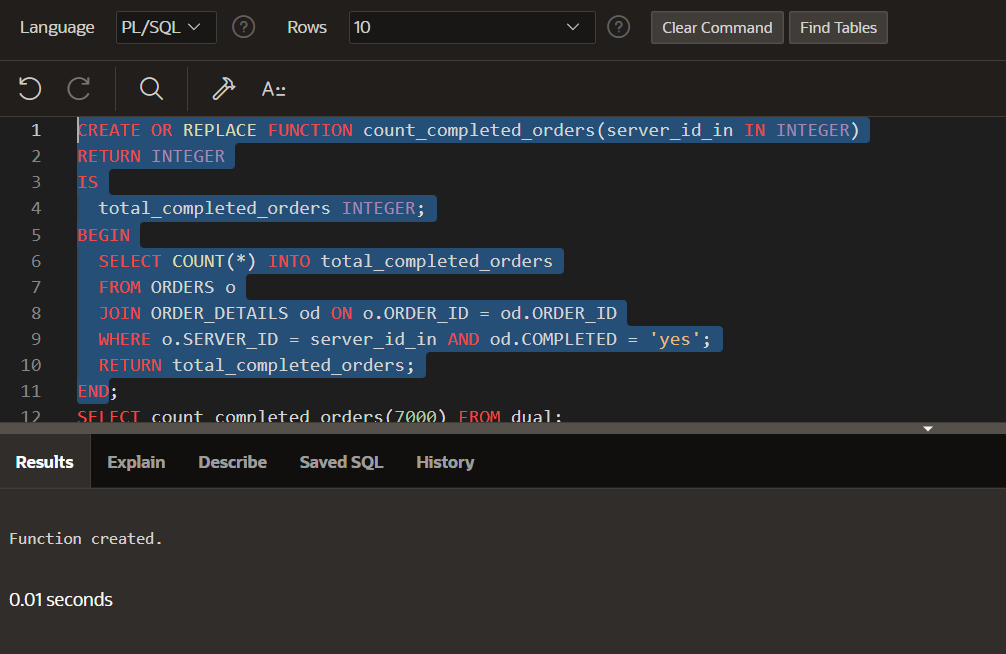
The **dbms\_output.put\_line** statement is used to display the text "Daily Sales Total:" in the console.

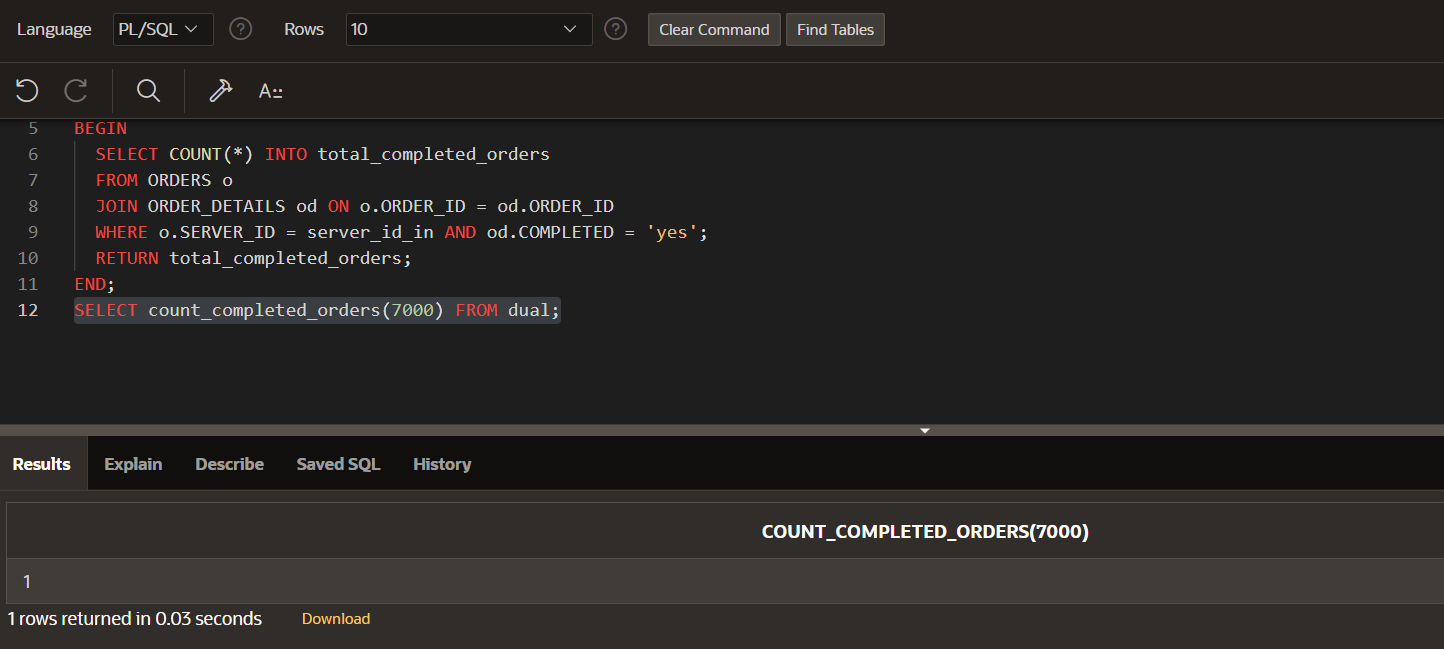
The loop iterates through each row returned by the cursor, assigning the values of the **date\_b** and **total\_sales** columns to the **daily\_sale** variable.

**Why do we need cursor?**

The use of a cursor is essential in these procedures because it allows us to retrieve the data we need from multiple tables and perform calculations on that data. Without a cursor, we would need to retrieve all the data at once and perform the calculations from memory, which could be more resource-intensive and potentially cause performance issues. The cursor allows us to retrieve the data one row at a time and perform the necessary calculations and comparisons.

4) Function which counts the number of records





Firstly, a function named **count\_completed\_orders** is replaced or created, it takes an **integer** input parameter named **server\_id\_in** and returns an **integer** value.

The **is** keyword starts the declaration of the function's body, which is enclosed in the **begin** and **end** keywords.

A **select** statement counts the number of completed orders for the specified **server\_id\_in**.

The **join** keyword combines the **orders** and **order\_details** tables on the **order\_id** column. The **where** clause filters the results to only include orders with a **server\_id** of **server\_id\_in** and **completed** value of **'yes'**.

The **count(\*)** function counts the number of rows that match the filtering conditions and stores the result in the **total\_completed\_orders** variable using the **into** keyword.

Lastly, the **count\_completed\_orders** function with a parameter value of **7000** and returns the result. The **from dual** clause is required in Oracle SQL to execute a function call without referencing a table.