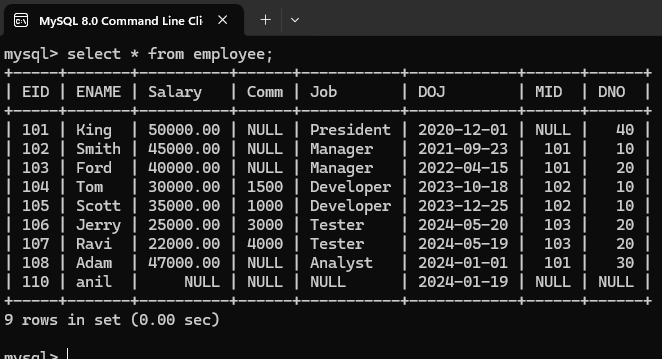
# RDBMS and SQL Assignment (part 2)

**Assignment 1:** Write a SELECT query to retrieve all columns from a 'Employee' table, and modify it to return only the employee name and job for employee in a specific salary range.

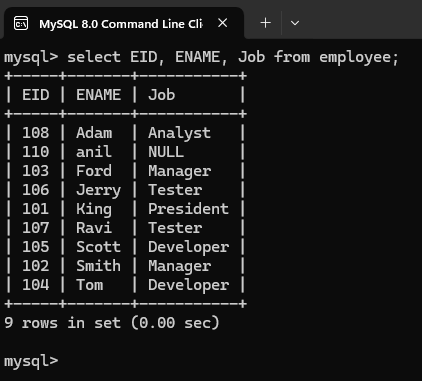
To retrieve all columns from a table we use select command.



To select or modify to return only specific columns we use the following query.

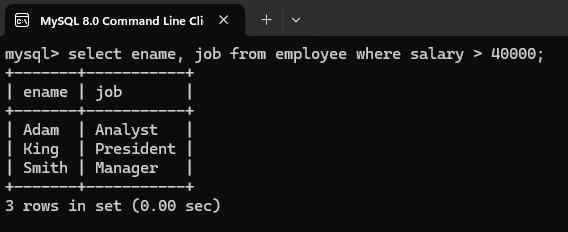
**Syntax: SELECT column1,column2 FROM table\_name;**

The above syntax returns only specified columns even the table consists of n number of columns.

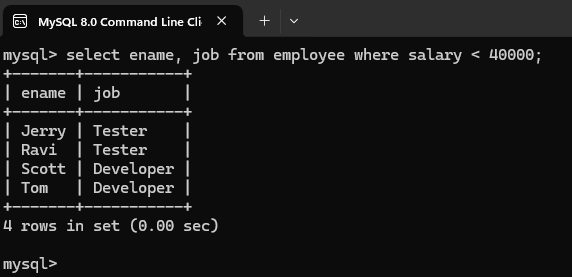


To return only some rows we use where clause with some condition, if the condition satisfies then the row will be displayed.

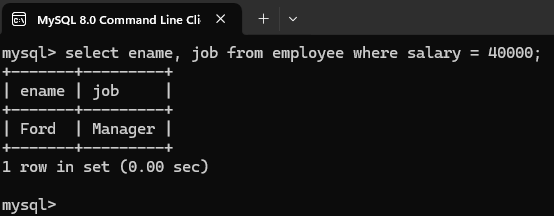
**Syntax: SELECT column1,coumn2 FROM table\_name where condition;**



For Salary less than 40000 we will get other details.



For salary equals to 40000 we will get rest details.

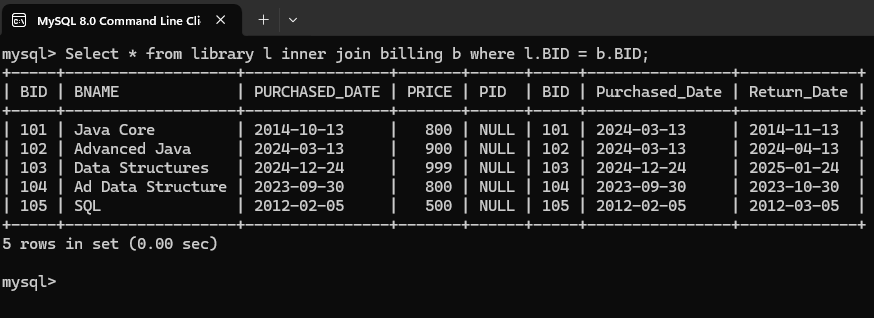


**Assignment 2:** Craft a query using an INNER JOIN to combine 'library' and 'billing' tables for books in a specified region, and a LEFT JOIN to display all data including those without orders.

**INNER JOIN:**

INNER JOIN Keyword is used to display or return all values from 2 tables which are matching.

**Syntax: SELECT \* FROM table\_name1 INNER JOIN table\_name2 where condition;**



If we don’t mention the condition the table\_1 will multiple with table\_2 and the result will be



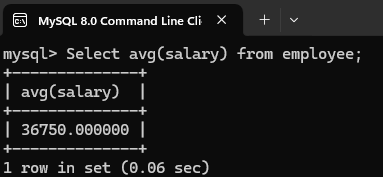
The above data is not the perfect data Since the library data will multiply with the billing data.

**Assignment 3:** Utilize a sub-query to find employees who have salary above the average salary value, and write a UNION query to combine two SELECT statements with the same number of columns.

**AVG:**

AVG returns average of all integers of a specified column, but the column should be integer.

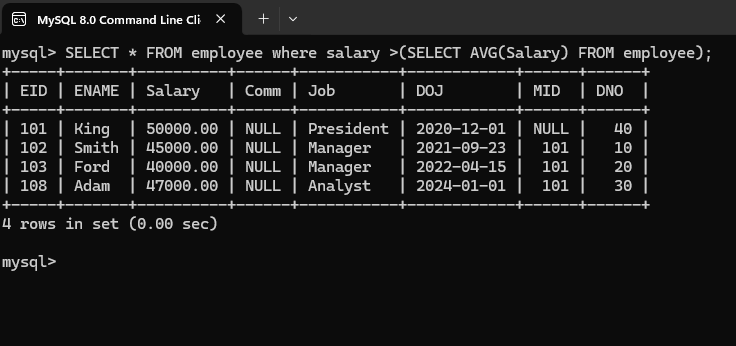
**Syntax: SELECT AVG(column\_name) FROM table\_name;**



**Sub query :**

Writing query inside a query is known as sub query.

**Syntax: CREATE TABLE table\_name as (sub query);**

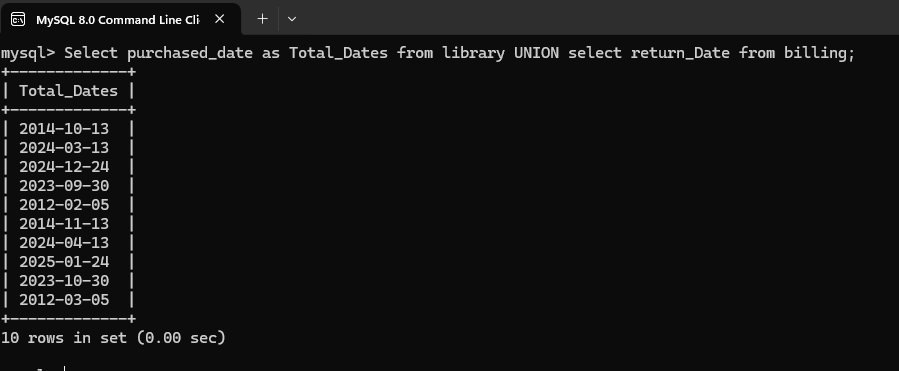


**UNION:**

UNION command is used for combining 2 tables if, but the table must contain common columns.

**Syntax: SELECT column from table\_name1 UNION SELECT column**

**from table\_name2;**



**Assignment 4:** Compose SQL statements to BEGIN a transaction, INSERT a new record into the 'library' table, COMMIT the transaction, then UPDATE the 'billing' table, and ROLLBACK the transaction.

**Transaction:**

Transaction in mysql is a group of statements refers to a single unit of work. Which can be revert back if needed, but only before commit.

**Syntax:**

**Start transaction**

**Statement\_1;**

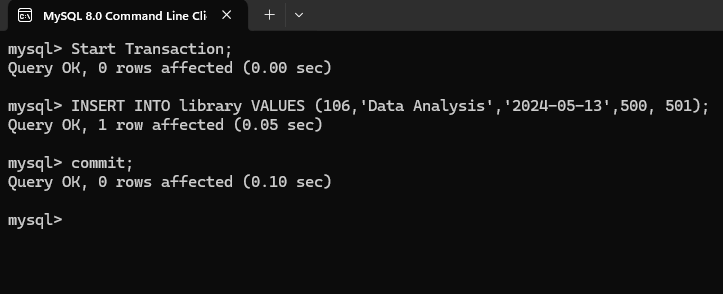
**Statement\_2;**

**:**

**:**

**Statement\_n;**

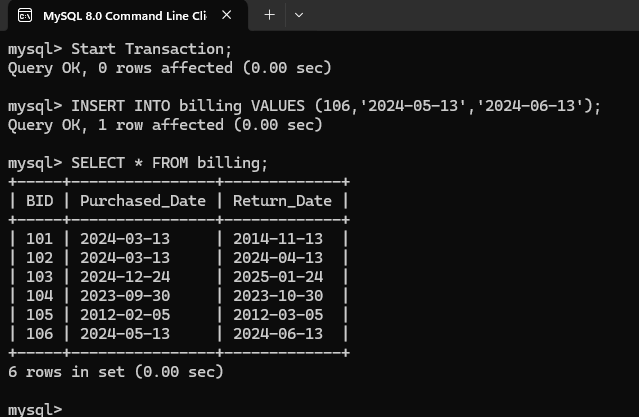
**Commit; //** if you don’t want to save then **rollback**



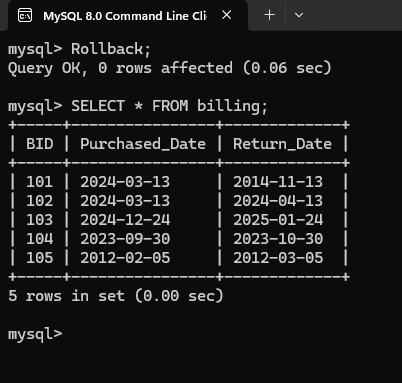
**Note: In the above statements if commit is not used then the statements are temporarily execute(if we close mysql then data will delete) to save the statements executed we need to use commit statement.**

**ROLLBACK:**

Rollback is used to rollback to the current transaction to the starting o the transaction.



**At this state if we use rollback all the statements in the transaction will get rollback.**



In the above query the inserted statements where got reverted.

**Assignment 5:** Begin a transaction, perform a series of INSERTs into 'library', setting a SAVEPOINT after each, rollback to the second SAVEPOINT, and COMMIT the overall transaction.

**SAVEPOINT:**

Savepoint is a logical rollback point where we can rollback to a certain point in a transaction.

**Syntax:**

**Start Transaction**

**Statement 1;**

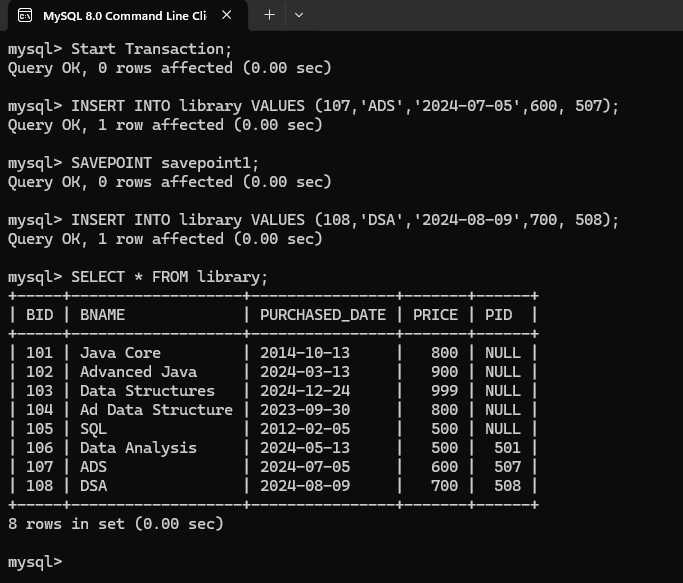
**SAVEPOINT identifier;**

**Statement 2;**

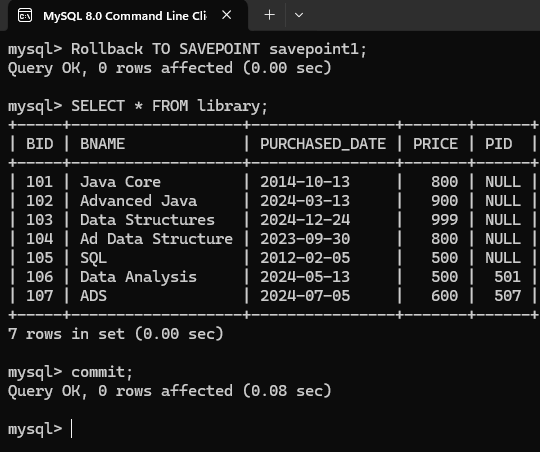
**:**

**:**

**Rollback to SAVEPOINT identifier;**



At this state if we use rollback to specific savepoint then till savepoint1 it will be rollback.



In the above query the rollback is happened till the savepoint not to the starting of transaction.

**\*NOTE:**

Commit must be used after the transaction even if we use savepoints.

**Assignment 6:** Draft a brief report on the use of transaction logs for data recovery and create a hypothetical scenario where a transaction log is instrumental in data recovery after an unexpected shutdown.

**Transaction Logs for Data Recovery:**

Transaction logs play a crucial role in ensuring data integrity and recoverability in database management systems (DBMS). In this report, we explore the purpose of transaction logs, their characteristics, and scenarios where they are instrumental in data recovery.

**Purpose of Transaction Logs**

1. **Recording Transactions**: Transaction logs record all modifications made to a database. These include updates, inserts, and deletes performed by transactions.
2. **Point-in-Time Recovery:** Transaction logs allow for point-in-time recovery. In case of system failures or disasters, we can restore the database to a specific point just before the issue occurred, minimizing data loss.

**Transaction Log Characteristics**

**Sequential Records:** The transaction log maintains a sequential record of all changes.

**Durability:** Log entries are durable and survive system crashes.

Circular Buffer: The log operates as a circular buffer, reusing space once it reaches its maximum size.

**Hypothetical Scenario:**

Unexpected Shutdown Recovery consider the following scenario:

**The Incident:** A power outage occurs, leading to an unexpected shutdown of the database server.

**Database State:** At the time of the shutdown, some transactions were incomplete, and not all modifications were written to the data files.

**Recovery Process:**

Upon server restart, the DBMS runs a recovery process for each database.

The transaction log is used to roll forward all modifications recorded in the log (i.e., apply changes to the data files).

Incomplete transactions are rolled back to maintain database integrity.

**Result:** The database is brought back to a consistent state, ensuring data consistency and reliability.

**Conclusion**

Transaction logs are essential for data recovery. They allow to reconstruct databases after system failures, ensuring that no data is lost and maintaining the integrity of the database.

**\*NOTE:** Never delete or move transaction logs unless fully understanding the ramifications.