DBMS VIVA questions

Experiment 1: Employee Table Creation and Basic Constraints

1. Q: What is the purpose of ROLLBACK command in SQL? A: ROLLBACK is used to undo transactions that haven't been saved to the database using COMMIT. It restores the database to the last committed state.
2. Q: Why do we need to grant permissions to users in a database? A: Permissions are granted to control database security by specifying which users can perform what operations (like SELECT, INSERT, UPDATE, DELETE) on specific database objects.
3. Q: What happens if you try to insert NULL values into a column with NOT NULL constraint? A: The insertion will fail and generate an error because NOT NULL constraint ensures that a column cannot have NULL values.
4. Q: What is the difference between PRIMARY KEY and NOT NULL constraint? A: PRIMARY KEY ensures both uniqueness and non-null values, while NOT NULL only ensures that values cannot be null but allows duplicates.
5. Q: Can a table have multiple PRIMARY KEYs? A: No, a table can have only one PRIMARY KEY constraint, though it can be composed of multiple columns (composite key).
6. Q: What is the syntax for adding a PRIMARY KEY constraint after table creation? A: ALTER TABLE table_name ADD CONSTRAINT constraint_name PRIMARY KEY (column_name);
7. Q: How does COMMIT differ from ROLLBACK? A: COMMIT permanently saves all transactions to the database, while ROLLBACK undoes all transactions since the last COMMIT.
8. Q: What happens to existing data when adding a NOT NULL constraint to a column? A: If the column contains NULL values, the ALTER TABLE statement will fail unless a default value is specified.
9. Q: Can you enforce both UNIQUE and NOT NULL constraints on a column? A: Yes, you can have both constraints on a column to ensure values are both unique and not null.
10. Q: What is the difference between dropping and truncating a table? A: DROP removes the table structure and data, while TRUNCATE removes only the data but keeps the table structure.

Experiment 2: Table Modification and Column Operations

1. Q: What is the purpose of ALTER command in SQL? A: ALTER is used to modify the structure of existing database objects, like adding/removing columns, modifying data types, or renaming columns.
2. Q: How do you add a new column to an existing table? A: Using ALTER TABLE table_name ADD column_name datatype;

3. Q: What happens to existing records when you add a new column? A: The new column is added with NULL values (or default values if specified) for all existing records.
4. Q: How can you rename a column in SQL? A: Using ALTER TABLE table_name RENAME COLUMN old_name TO new_name;
5. Q: What is the difference between DELETE and TRUNCATE? A: DELETE can be used with WHERE clause to remove specific rows and can be rolled back, while TRUNCATE removes all rows and cannot be rolled back.
6. Q: Can you change the data type of a column containing data? A: Yes, but the conversion must be compatible with existing data, or it will fail.
7. Q: How do you update multiple records simultaneously? A: Using UPDATE table_name SET column=value WHERE condition;
8. Q: What happens if you update a record without WHERE clause? A: All records in the table will be updated with the new value.
9. Q: How can you modify multiple columns in a single ALTER statement? A: By separating multiple ADD, MODIFY, or DROP clauses with commas.
10. Q: What is the difference between WHERE and HAVING clause? A: WHERE filters individual rows before grouping, while HAVING filters groups after GROUP BY clause.

Experiment 3: Aggregate Functions and Grouping

1. Q: What is the difference between COUNT(*) and COUNT(column_name)? A: COUNT(*) counts all rows including NULL values, while COUNT(column_name) counts only non-NULL values in that column.
2. Q: Can you use WHERE clause with GROUP BY? If yes, what is the order? A: Yes, WHERE comes before GROUP BY. The order is: WHERE -> GROUP BY -> HAVING -> ORDER BY.
3. Q: Why can't we use aggregate functions in WHERE clause? A: Aggregate functions work on groups of rows, while WHERE filters individual rows before grouping occurs.
4. Q: What's the difference between ORDER BY and GROUP BY? A: ORDER BY sorts the result set, while GROUP BY combines rows with same values into summary rows.
5. Q: Can you use multiple aggregate functions in a single query? A: Yes, you can use multiple aggregate functions like SELECT COUNT(*), SUM(salary), AVG(age) FROM employee;
6. Q: What happens if you GROUP BY multiple columns? A: Records are grouped based on unique combinations of all specified columns.
7. Q: Can MIN and MAX functions be used with non-numeric data? A: Yes, they work with any comparable data types including strings and dates.
8. Q: What is the purpose of HAVING clause? A: HAVING filters groups after GROUP BY, similar to how WHERE filters rows.
9. Q: How does AVG function handle NULL values? A: AVG ignores NULL values in its calculation.

10. Q: Can you use ORDER BY with columns not in SELECT list? A: Yes, but it's not recommended as it can make queries less readable.

Experiment 4: Triggers

1. Q: What is a trigger and when does it fire? A: A trigger is a special type of stored procedure that automatically executes when a specified event (INSERT, UPDATE, DELETE) occurs.
2. Q: What's the difference between row-level and statement-level triggers? A: Row-level triggers fire once for each row affected, while statement-level triggers fire once per SQL statement.
3. Q: What are OLD and NEW references in triggers? A: OLD refers to the row's values before update/delete, NEW refers to the new values being inserted/updated.
4. Q: Can a trigger call itself recursively? A: No, triggers cannot be recursive to prevent infinite loops.
5. Q: What's the difference between BEFORE and AFTER triggers? A: BEFORE triggers fire before the triggering action, AFTER triggers fire after the action completes.
6. Q: Can you have multiple triggers on same event? A: Yes, but you need to specify their firing order.
7. Q: What happens if a trigger throws an error? A: The entire transaction is rolled back, including the triggering statement.
8. Q: Can triggers modify tables other than the one they're defined on? A: Yes, triggers can modify other tables.
9. Q: What is mutating table error in triggers? A: It occurs when a trigger attempts to query or modify the same table that triggered it.
10. Q: How can you disable a trigger? A: Using ALTER TRIGGER trigger_name DISABLE;

Experiment 5: Cursors

1. Q: What is a cursor and why do we use it? A: A cursor is a database object that allows row-by-row processing of query results. It's used when we need to perform operations on each row individually.
2. Q: What are the types of cursors in PL/SQL? A: There are two types: Implicit cursors (automatically created by Oracle) and Explicit cursors (declared and controlled by programmer).
3. Q: What are the steps involved in using a cursor? A: The steps are: DECLARE cursor, OPEN cursor, FETCH data, process data, and CLOSE cursor.
4. Q: What happens if you try to fetch from a closed cursor? A: It will raise an INVALID_CURSOR exception.
5. Q: What is cursor_status attribute used for? A: It returns the status of the cursor: OPEN, CLOSED, or INVALID.

6. Q: How do you handle NO_DATA_FOUND exception in cursors? A: Using exception handling block to catch NO_DATA_FOUND when FETCH returns no rows.
7. Q: What is a cursor FOR loop? A: It's a simplified way to process cursor results without explicit FETCH statements.
8. Q: Can multiple users access the same cursor simultaneously? A: No, cursors are session-specific and cannot be shared between sessions.
9. Q: What happens if you don't close a cursor? A: It continues to consume system resources until the session ends.
10. Q: What is the difference between %FOUND and %NOTFOUND cursor attributes? A: %FOUND returns TRUE if the last fetch got a row, %NOTFOUND returns TRUE if the last fetch failed to get a row.

Experiment 6: Parameterized Cursors and Merging

1. Q: What is a parameterized cursor? A: It's a cursor that accepts parameters, making it more flexible and reusable.
2. Q: How do you declare a parameterized cursor? A: CURSOR cursor_name (parameter_name datatype) IS SELECT statement;
3. Q: What is the MERGE statement used for? A: MERGE combines INSERT and UPDATE operations in a single statement based on a condition.
4. Q: How does MERGE handle duplicate records? A: It can either update existing records or skip them based on the MERGE conditions specified.
5. Q: Can you use multiple conditions in MERGE statement? A: Yes, you can use multiple conditions in the ON clause of MERGE.
6. Q: What happens if MERGE encounters an error? A: The entire MERGE operation is rolled back if any error occurs.
7. Q: Can you use subqueries in MERGE statement? A: Yes, subqueries can be used in the ON clause and UPDATE/INSERT sections.
8. Q: What is the advantage of using MERGE over separate INSERT/UPDATE? A: MERGE is more efficient as it requires only one pass through the data.
9. Q: Can you use WHERE clause in MERGE statement? A: Yes, you can use WHERE clause in the INSERT and UPDATE parts of MERGE.
10. Q: What is the difference between MERGE and INSERT ALL? A: MERGE can update existing records while INSERT ALL only inserts new records.

Experiment 7: MongoDB CRUD Operations

1. Q: What are the basic CRUD operations in MongoDB? A: Create (insert), Read (find), Update (update/updateMany), and Delete (remove/deleteMany).
2. Q: How do you insert a document in MongoDB? A: Using db.collection.insertOne() or db.collection.insertMany().

3. Q: What is the difference between update() and updateOne()? A: update() can modify multiple documents while updateOne() modifies only the first matching document.
4. Q: How do you query specific fields in MongoDB? A: Using projection in find(): db.collection.find({}, {field: 1}).
5. Q: What is the purpose of $set operator? A: $set modifies specific fields without affecting other fields in the document.
6. Q: How do you delete all documents in a collection? A: Using db.collection.deleteMany({}).
7. Q: What is the difference between drop() and remove()? A: drop() removes the entire collection while remove() deletes specific documents.
8. Q: How do you sort results in MongoDB? A: Using sort(): db.collection.find().sort({field: 1}) for ascending order.
9. Q: What is the purpose of limit() in MongoDB? A: limit() restricts the number of documents returned by a query.
10. Q: How do you perform a case-insensitive search in MongoDB? A: Using $regex with 'i' option: {field: {$regex: /pattern/i}}.