[Exploratory Data Analysis (EDA) Using SQL] {CheatSheet}

1. Basic Data Overview

- **Show Tables**: `SHOW TABLES;`
- **Describe Table Structure**: `DESCRIBE table_name;`
- **Select All Data from a Table**: `SELECT * FROM table_name;`
- **Count Rows in a Table**: `SELECT COUNT(*) FROM table_name;`
- **List Unique Values in a Column**: `SELECT DISTINCT column_name FROM table_name;`

2. Aggregations and Summaries

- **Count Distinct Values**: `SELECT COUNT(DISTINCT column_name) FROM table_name;`
- **Calculate Average Value**: `SELECT AVG(column_name) FROM table_name;`
- **Sum Values**: `SELECT SUM(column_name) FROM table_name;`
- **Find Maximum and Minimum**: `SELECT MAX(column_name), MIN(column_name) FROM table_name;`
- **Group By and Aggregate**: `SELECT column1, COUNT(*), AVG(column2) FROM table_name GROUP BY column1;`

3. Data Slicing and Filtering

- **Select with Specific Criteria**: `SELECT * FROM table_name WHERE condition;`
- **Filtering with Multiple Conditions**: `SELECT * FROM table_name WHERE condition1 AND condition2;`
- **Select with ORDER BY**: `SELECT * FROM table_name ORDER BY column ASC/DESC;`
- **Select with LIMIT**: `SELECT * FROM table_name LIMIT number;`
- **Using BETWEEN for Range**: `SELECT * FROM table_name WHERE column BETWEEN value1 AND value2;`

4. Working with Dates

- **Selecting Date Range**: `SELECT * FROM table_name WHERE date_column BETWEEN '2021-01-01' AND '2021-12-31';`

By: Waleed Mousa

- **Extract Year, Month, Day**: `SELECT YEAR(date_column), MONTH(date_column), DAY(date_column) FROM table_name;`
- **Date Format Conversion**: `SELECT DATE_FORMAT(date_column, '%Y-%m-%d') FROM table_name;`
- **Age Calculation from Birthdate**: `SELECT DATEDIFF(CURDATE(), birthdate_column) FROM table_name;`
- **Group By Year or Month**: `SELECT YEAR(date_column), COUNT(*) FROM table_name GROUP BY YEAR(date_column);`

5. String Operations

- **Concatenation of Strings**: `SELECT CONCAT(string1, string2) FROM table_name;`
- **String Length**: `SELECT LENGTH(string_column) FROM table_name;`
- **Substring Extraction**: `SELECT SUBSTRING(string_column, start, length) FROM table_name;`
- **Changing Case**: `SELECT UPPER(string_column), LOWER(string_column) FROM table_name;`
- **Finding String Position**: `SELECT INSTR(string_column, 'substring') FROM table_name;`

6. Conditional Logic

- **CASE Statement**: `SELECT CASE WHEN condition THEN 'result1' ELSE 'result2' END FROM table_name;`
- **IF Statement**: `SELECT IF(condition, 'result1', 'result2') FROM table_name;`
- **NULL Handling with COALESCE**: `SELECT COALESCE(column, 'default_value') FROM table_name;`
- **Conditional Aggregation**: `SELECT SUM(CASE WHEN condition THEN 1 ELSE 0 END) FROM table_name;`

7. Joins and Relationships

- **Inner Join**: `SELECT * FROM table1 INNER JOIN table2 ON table1.common_column = table2.common_column;`
- **Left Join**: `SELECT * FROM table1 LEFT JOIN table2 ON table1.common_column = table2.common_column;`

By: Waleed Mousa

- **Right Join**: SELECT * FROM table1 RIGHT JOIN table2 ON table1.common_column = table2.common_column;
- **Full Outer Join**: SELECT * FROM table1 FULL OUTER JOIN table2 ON table1.common_column = table2.common_column;
- **Cross Join**: SELECT * FROM table1 CROSS JOIN table2;

8. Subqueries and Nested Queries

- **Subquery in SELECT**: SELECT column, (SELECT AVG(column) FROM table) AS average FROM table;
- **Subquery in FROM**: SELECT * FROM (SELECT * FROM table) AS subtable;
- **Subquery in WHERE**: SELECT * FROM table WHERE column IN (SELECT column FROM another_table);

9. Data Cleaning

- **Removing Duplicates**: SELECT DISTINCT * FROM table_name;
- **Replacing NULL with Default Value**: SELECT IFNULL(column, 'default') FROM table_name;
- **Trimming Whitespaces**: SELECT TRIM(column) FROM table_name;
- **Handling Missing Data (Filter)**: SELECT * FROM table_name WHERE column IS NOT NULL;

10. Advanced Aggregation

- **Rollup for Subtotals**: SELECT column1, column2, SUM(column3) FROM table_name GROUP BY column1, column2 WITH ROLLUP;
- **Grouping Sets for Custom Aggregates**: SELECT column1, column2, SUM(column3) FROM table_name GROUP BY GROUPING SETS ((column1), (column2));
- **Window Functions for Running Totals**: SELECT column, SUM(column) OVER (ORDER BY column) FROM table_name;
- **Ranking within Groups**: SELECT column, RANK() OVER (PARTITION BY column1 ORDER BY column2) FROM table_name;

11. Performance and Optimization

- **Index Creation for Performance**: CREATE INDEX idx_column ON table_name (column);

By: Waleed Mousa

- **Using EXPLAIN for Query Analysis**: `EXPLAIN SELECT * FROM table_name;`
- **Optimizing with Query Hints**: `SELECT /*+ HINT */ * FROM table_name;`
- **Batch Processing with LIMIT and OFFSET**: `SELECT * FROM table_name LIMIT 1000 OFFSET 1000;`

12. Data Export/Import

- **Exporting Data to CSV**: `SELECT * INTO OUTFILE '/path/to/file.csv' FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"' LINES TERMINATED BY '\n' FROM table_name;`
- **Importing Data from CSV**: `LOAD DATA INFILE '/path/to/file.csv' INTO TABLE table_name FIELDS TERMINATED BY ',' ENCLOSED BY '"' LINES TERMINATED BY '\n';`

13. Database and Table Management

- **Create New Database**: `CREATE DATABASE database_name;`
- **Drop Database**: `DROP DATABASE database_name;`
- **Create Table with Specific Structure**: `CREATE TABLE table_name (column1 datatype, column2 datatype);`
- **Modify Existing Table Structure**: `ALTER TABLE table_name ADD/DROP/MODIFY column_name datatype;`
- **Delete Table**: `DROP TABLE table_name;`

14. Working with Different Data Types

- **Casting Data Types**: `SELECT CAST(column AS datatype) FROM table_name;`
- **Working with Enums**: `SELECT * FROM table_name WHERE enum_column = 'value1';`
- **Handling JSON Data**: `SELECT json_extract(json_column, '$.key') FROM table_name;`
- **Manipulating Geospatial Data**: `SELECT ST_Distance(geo_column, ST_GeomFromText('POINT(lat lon)')) FROM table_name;`

15. Analyzing Text and Patterns

By: Waleed Mousa

- **LIKE Operator for Pattern Matching**: `SELECT * FROM table_name WHERE column LIKE '%pattern%';`
- **Regular Expressions**: `SELECT * FROM table_name WHERE column REGEXP 'regexp_pattern';`
- **Splitting and Extracting from Strings**: `SELECT SUBSTRING_INDEX(column, 'delimiter', part) FROM table_name;`

16. Advanced Joins

- **Self Join for Hierarchical Data**: `SELECT t1.column, t2.column FROM table t1 JOIN table t2 ON t1.id = t2.parent_id;`
- **Join with Aggregation**: `SELECT t1.column, SUM(t2.column) FROM table1 t1 JOIN table2 t2 ON t1.id = t2.foreign_id GROUP BY t1.column;`
- **Join with Complex Conditions**: `SELECT * FROM table1 t1 JOIN table2 t2 ON t1.id = t2.foreign_id AND t2.condition = 'value';`
- **Lateral Join (Correlated Subquery)**: `SELECT * FROM table1 t1, LATERAL (SELECT * FROM table2 t2 WHERE t2.foreign_id = t1.id) as subquery;`

17. Database Views

- **Create View for Reuse**: `CREATE VIEW view_name AS SELECT column1, column2 FROM table_name WHERE condition;`
- **Querying a View**: `SELECT * FROM view_name;`
- **Updating View Definition**: `CREATE OR REPLACE VIEW view_name AS SELECT column1 FROM table_name;`
- **Dropping a View**: `DROP VIEW view_name;`

18. Data Integrity and Constraints

- **Creating Table with Constraints**: `CREATE TABLE table_name (column1 datatype PRIMARY KEY, column2 datatype UNIQUE);`
- **Adding Foreign Key Constraint**: `ALTER TABLE child_table ADD FOREIGN KEY (foreign_key_column) REFERENCES parent_table (parent_key_column);`
- **Enforcing Data Integrity with Check**: `ALTER TABLE table_name ADD CHECK (condition);`

By: Waleed Mousa

- **Validating Constraints**: SELECT * FROM table_name WHERE NOT VALID condition;
- **Creating Unique Constraints**: ALTER TABLE table_name ADD UNIQUE (column);

19. Transaction Control

- **Start a Transaction**: START TRANSACTION;
- **Commit a Transaction**: COMMIT;
- **Rollback a Transaction**: ROLLBACK;
- **Set Transaction Isolation Level**: SET TRANSACTION ISOLATION LEVEL READ COMMITTED;

20. Advanced Subqueries

- **Correlated Subquery**: SELECT * FROM table1 t1 WHERE EXISTS (SELECT * FROM table2 t2 WHERE t1.id = t2.foreign_id);
- **Subquery as a Table**: SELECT * FROM (SELECT * FROM table) AS sub;
- **Using Subquery in SELECT Clause**: SELECT id, (SELECT COUNT(*) FROM table2 WHERE foreign_id = table1.id) FROM table1;

21. Working with Large Datasets

- **Batch Deletion to Avoid Locks**: DELETE FROM table_name WHERE condition LIMIT 1000;
- **Efficient Pagination**: SELECT * FROM table_name ORDER BY id LIMIT 1000 OFFSET 5000;
- **Optimized Aggregation for Large Tables**: SELECT APPROX_COUNT_DISTINCT(column) FROM big_table;

22. Data Warehousing Commands

- **Creating Fact and Dimension Tables**: CREATE TABLE fact_table (key INT, measure INT);
- **Querying Star Schema**: SELECT * FROM fact_table JOIN dimension_table ON fact_table.dim_key = dimension_table.key;
- **ETL Operations**: INSERT INTO table SELECT * FROM external_source;

By: Waleed Mousa

23. Working with Indexes

- **Creating an Index**: `CREATE INDEX idx_column ON table_name (column);`
- **Using Index Hint**: `SELECT * FROM table_name USE INDEX (idx_column);`
- **Dropping an Index**: `DROP INDEX idx_column ON table_name;`

24. Database Optimization

- **Analyzing Table for Optimization**: `ANALYZE TABLE table_name;`
- **Optimizing Table**: `OPTIMIZE TABLE table_name;`
- **Database Normalization**: `SELECT * INTO new_table FROM (SELECT DISTINCT column FROM table_name) AS temp;`

25. Query Performance Analysis

- **Query Execution Plan**: `EXPLAIN SELECT * FROM table_name WHERE condition;`
- **Monitoring Database Performance**: `SHOW STATUS LIKE 'Key%';`
- **Identifying Long-running Queries**: `SHOW PROCESSLIST;`

26. Backup and Recovery

- **Backing up a Database**: `mysqldump -u user -p database_name > backup.sql;`
- **Restoring from a Backup**: `mysql -u user -p database_name < backup.sql;`