SoftNexis

Task 1: Basic Data Retrieval & Filtering - Complete Breakdown

Description

New SQL developers will learn foundational data extraction techniques using single database tables. This task focuses on querying structured data using core `SELECT` statements with filtering, sorting, and pattern-matching operations - essential skills for 92% of data-related roles (StackOverflow Survey 2023).

Responsibilities

1. Column Selection
    - Retrieve specific columns instead of entire tables (`SELECT col1, col2` vs `SELECT *`)
    - Handle calculated columns (e.g., `SELECT salary * 1.1 AS new_salary`)
2. Precision Filtering
    - Apply `WHERE` clauses with:
        - Equality operators (`=`, `!=`)
        - Numerical comparisons (`>`, `<`, `>=`, `<=`)
        - Range operators (`BETWEEN` for numbers/dates)
        - Text pattern matching (`LIKE`, `NOT LIKE` with `%` and `_` wildcards)
    - Combine conditions with `AND`/`OR`
    - Handle `NULL` values (`IS NULL`/`IS NOT NULL`)
3. Data Organization
    - Sort results using `ORDER BY` (single/multiple columns)
    - Control sort direction (`ASC` for ascending, `DESC` for descending)
    - Implement combined sorts (e.g., `ORDER BY department ASC, salary DESC`)
4. Output Control
    - Limit results with `LIMIT` (MySQL/PostgreSQL) or `TOP` (SQL Server)
    - Paginate results using `OFFSET`
5. Data Validation
    - Verify query accuracy through record counts and sample checks
    - Compare output against source data integrity

Skills Gained

✅ Core SQL Syntax Mastery

- Correct clause sequencing (`SELECT → FROM → WHERE → ORDER BY`)
- Syntax error troubleshooting (e.g., missing commas, quote mismatches)

✅ Data Profiling Competence

- Identify data types (dates vs strings, integers vs floats)
- Detect anomalies (NULL values, outliers, formatting inconsistencies)

✅ Business-Ready Outputs

- Structure results for stakeholder consumption
- Format dates/numbers for readability (e.g., `DATE_FORMAT(hire_date, '%Y-%m')`)

✅ Performance Awareness

- Avoid `SELECT *` to reduce memory usage
- Use `LIMIT` on large tables during exploration
- Recognize expensive operations (e.g., `LIKE '%text'` without indexes)

✅ Real-World Problem Solving

- Translate business questions into queries

  "Show marketing hires in 2023 with salaries >$50K" →

```
SELECT first_name, hire_date, salary
FROM employees
WHERE department = 'Marketing'
  AND hire_date >= '2023-01-01'
```

- ```
    AND salary > 50000
  ```

  ```

Practical Dataset & Validation

Recommended Dataset: Employees Database (MySQL)

- Tables Used: `employees` (300k+ rows), `salaries`
- Validation Query:

```
-- Verify task mastery
SELECT
  COUNT(*) AS total_filtered,
  MIN(salary) AS min_salary,
  MAX(hire_date) AS latest_hire
FROM employees
WHERE
  department = 'Sales'
  AND hire_date BETWEEN '1990-01-01' AND '1999-12-31'
```

- `  AND last_name LIKE 'S%'`

Output Checklist:

Metric	Expected Result	Intern's Result	Pass/Fail
Row count	142 (example)	[ ]	☐
Min salary	42,000	[ ]	☐
Latest hire	1999-12-24	[ ]	☐

Progression Metrics

Intern Success Signals:

🟢 Completes 10+ varied queries with <20% syntax error rate

🟢 Explains why `WHERE salary > 50000` returns different results than `WHERE salary`

`>= 50000`

🟢 Recognizes when `BETWEEN` is inclusive vs exclusive

Common Failure Points:

🔴 Mixing `AND`/`OR` without parentheses

🔴 Case sensitivity mismatches (`'sales'` vs `'Sales'`)

🔴 Date format errors (`'01-01-2023'` vs `'2023-01-01'`)

💡 Expert Tip: "Always start with `SELECT COUNT(*)` to validate filters before fetching full data. Saves 70% of debugging time in production."

- Senior DBA, Financial Services Industry

Next Task: Task 2: Data Aggregation & Reporting →

Foundation → Reporting progression based on AWS Data Analytics competency paths