# Web app outline and role

Got it 👍 You want a **demo version of the stored procedures in MySQL** for the scenarios you listed.
Since MySQL doesn't use PL/pgSQL like PostgreSQL, we'll write them in **MySQL's procedure syntax**.

Let's assume your table is named `results` with columns like:

```sql
results(
    student_id INT,
    regulation VARCHAR(10),
    academic_year INT,
    course_id VARCHAR(20),
    subject_id VARCHAR(20),
    grade VARCHAR(5),
    pass_fail VARCHAR(10),   -- 'Pass' / 'Fail'
    mode_of_study VARCHAR(20) -- 'Online' / 'Distance'
);
```

---

## 🔹 Demo Stored Procedures in MySQL

### 1. Regulation-Year Summary

```sql
DELIMITER $$

CREATE PROCEDURE sp_regulation_year_summary(IN reg_id VARCHAR(10), IN acad_year INT)
BEGIN
    SELECT
        course_id,
        COUNT(*) AS total_students,
        SUM(CASE WHEN pass_fail = 'Pass' THEN 1 ELSE 0 END) AS pass_count,
        SUM(CASE WHEN pass_fail = 'Fail' THEN 1 ELSE 0 END) AS fail_count,
        ROUND(SUM(CASE WHEN pass_fail = 'Pass' THEN 1 ELSE 0 END) * 100.0 / COUNT(*), 2) AS pass_percentage
```

```
    FROM results
    WHERE regulation = reg_id AND academic_year = acad_year
    GROUP BY course_id;
END$$

DELIMITER ;
```

---

## 2. Course Performance Trend

sql

```
DELIMITER $$

CREATE PROCEDURE sp_course_trend(IN c_id VARCHAR(20))
BEGIN
    SELECT
        academic_year,
        ROUND(SUM(CASE WHEN pass_fail = 'Pass' THEN 1 ELSE 0 END) * 100.0 / COUNT(*), 2)
AS pass_percentage
    FROM results
    WHERE course_id = c_id
    GROUP BY academic_year
    ORDER BY academic_year;
END$$

DELIMITER ;
```

---

## 3. Subject Difficulty Index

sql

```
DELIMITER $$

CREATE PROCEDURE sp_subject_difficulty(IN reg_id VARCHAR(10), IN acad_year INT, IN c_id
VARCHAR(20))
BEGIN
    SELECT
        subject_id,
        ROUND(SUM(CASE WHEN pass_fail = 'Fail' THEN 1 ELSE 0 END) * 100.0 / COUNT(*), 2) AS
```

```
fail_rate
    FROM results
    WHERE regulation = reg_id AND academic_year = acad_year AND course_id = c_id
    GROUP BY subject_id
    ORDER BY fail_rate DESC;
END$$

DELIMITER ;
```

---

## 4. Mode of Study Comparison

sql

```
DELIMITER $$

CREATE PROCEDURE sp_mode_comparison(IN reg_id VARCHAR(10), IN acad_year INT, IN c_id
VARCHAR(20))
BEGIN
    SELECT
        mode_of_study,
        ROUND(SUM(CASE WHEN pass_fail = 'Pass' THEN 1 ELSE 0 END) * 100.0 / COUNT(*), 2)
AS pass_percentage
    FROM results
    WHERE regulation = reg_id AND academic_year = acad_year AND course_id = c_id
    GROUP BY mode_of_study;
END$$

DELIMITER ;
```

---

## 5. Top/Bottom Performing Students

sql

```
DELIMITER $$

CREATE PROCEDURE sp_student_rank(IN reg_id VARCHAR(10), IN acad_year INT, IN c_id
VARCHAR(20))
BEGIN
    SELECT
```

```
    student_id,
    SUM(CASE WHEN pass_fail = 'Pass' THEN 1 ELSE 0 END) AS subjects_passed,
    COUNT(*) AS total_subjects,
    ROUND(SUM(CASE WHEN pass_fail = 'Pass' THEN 1 ELSE 0 END) * 100.0 / COUNT(*), 2)
AS pass_percentage
  FROM results
  WHERE regulation = reg_id AND academic_year = acad_year AND course_id = c_id
  GROUP BY student_id
  ORDER BY pass_percentage DESC;
END$$

DELIMITER ;
```

---

## 6. Grade Distribution per Subject

sql

```
DELIMITER $$

CREATE PROCEDURE sp_grade_distribution(IN reg_id VARCHAR(10), IN acad_year INT, IN
subj_id VARCHAR(20))
BEGIN
  SELECT
    grade,
    COUNT(*) AS student_count
  FROM results
  WHERE regulation = reg_id AND academic_year = acad_year AND subject_id = subj_id
  GROUP BY grade
  ORDER BY grade;
END$$

DELIMITER ;
```

---

## 7. Multi-Year Trend

sql

```
DELIMITER $$
```

```
CREATE PROCEDURE sp_multi_year_summary(IN reg_id VARCHAR(10))
BEGIN
  SELECT
    academic_year,
    ROUND(SUM(CASE WHEN pass_fail = 'Pass' THEN 1 ELSE 0 END) * 100.0 / COUNT(*), 2)
AS pass_percentage
  FROM results
  WHERE regulation = reg_id
  GROUP BY academic_year
  ORDER BY academic_year;
END$$

DELIMITER ;
```

---

## 🔹 How to Run These

1. Create your `results` table and insert data (from Excel/PDF after parsing).
2. Copy each procedure into MySQL Workbench or CLI.
3. Call them like:

```sql
CALL sp_regulation_year_summary('R2019', 2024);
CALL sp_course_trend('CSE101');
CALL sp_subject_difficulty('R2019', 2024, 'CSE101');
CALL sp_mode_comparison('R2019', 2024, 'CSE101');
CALL sp_student_rank('R2019', 2024, 'CSE101');
CALL sp_grade_distribution('R2019', 2024, 'MATH101');
CALL sp_multi_year_summary('R2019');
```

---

👉 This set gives you **all backend logic for analytics**.
Frontend (React/Flask) can just call these via API → return JSON → visualize.

---

Do you want me to also give you a **sample Python (Flask) API wrapper** that connects to MySQL and calls these procedures (so you can hand it to the visualization team)?