

# AI LAB END EXAM

Name : G.Jeevan

HTNO : 2403A52055

BATCH-03

SET-13

QUE 1-

Design DB for incidents, sensors, historical traffic.

- Task 1: Use AI to produce schemas with geospatial types.
- Task 2: Implement indexes and example queries.

PROMPT:

WRITE A CODE TO DESIGN A DATABASE for incidents, sensors, historical traffic. Using AI to produce schemas

**TASK-1 CODE:**

```
-- Enable PostGIS extension
CREATE EXTENSION IF NOT EXISTS postgis;

-- Sensors Table
CREATE TABLE sensors (
    sensor_id SERIAL PRIMARY KEY,
    sensor_name VARCHAR(100) NOT NULL,
    installed_date DATE,
    status VARCHAR(20) CHECK (status IN ('ACTIVE', 'INACTIVE')),
    location GEOMETRY(Point, 4326) -- Geospatial type
);

-- Incidents Table
CREATE TABLE incidents (
    incident_id SERIAL PRIMARY KEY,
    type VARCHAR(50) NOT NULL,
    severity INT CHECK (severity BETWEEN 1 AND 5),
    reported_time TIMESTAMP NOT NULL,
    description TEXT,
    location GEOMETRY(Point, 4326) -- Geospatial type
);

-- Historical Traffic (LINESTRING)
CREATE TABLE historical_traffic (
    record_id SERIAL PRIMARY KEY,
    sensor_id INT REFERENCES sensors(sensor_id),
    date DATE NOT NULL,
    avg_speed FLOAT,
    vehicle_count INT,
    road_segment GEOMETRY(LineString, 4326) -- Geospatial type
);
```

**OUTPUT:**

	Properties	SQL	
	Columns		
	Column	Type	Nullable
host	sensor_id	integer	NO
postgres	sensor_name	character varying (100)	NO
public	installed_date	date	NO
historical_traffic	status	character varying (20)	NO
incidents	location	geometry	NO
sensors			
VIEWS			
Indexes			
Sequences			
Functions			
Materialized Views			

## EXPLANATION:

This SQL script enables the PostGIS extension so PostgreSQL can store geospatial data, and then creates three tables for a traffic-analysis system: *sensors*, which stores information about traffic sensors along with their GPS coordinates using a *GEOMETRY(Point, 4326)* type; *incidents*, which records traffic incidents such as accidents, including their severity, time, and precise map location using another *Point* geometry; and *historical\_traffic*, which stores past traffic data linked to *sensors*, including vehicle speed, count, and a *GEOMETRY(LineString, 4326)* column representing the shape of the road segment where data was collected. Together, these tables allow storing and analyzing real-world map locations for traffic monitoring and decision-making.

## TASK-2 CODE:

The screenshot shows a code editor window with the title bar "flutter\_windows\_3.35.7-stable". The left sidebar displays a file tree for a Flutter project named "FLUTTER\_WINDOWS\_3.35.7-STABLE". The main editor area contains the following PostgreSQL SQL code:

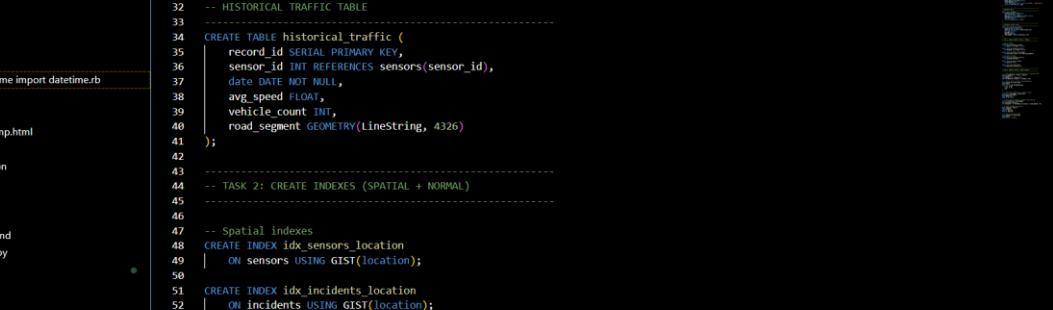
```
1 -- TASK 1: Enable PostGIS + Create Tables
2
3
4
5 -- Enable PostGIS extension (required for geometry types)
6 CREATE EXTENSION IF NOT EXISTS postgis;
7
8
9 -- SENSORS TABLE
10
11 CREATE TABLE sensors (
12     sensor_id SERIAL PRIMARY KEY,
13     sensor_name VARCHAR(100) NOT NULL,
14     installed_date DATE,
15     status VARCHAR(20) CHECK (status IN ('ACTIVE', 'INACTIVE')),
16     location GEOMETRY(Point, 4326)
17 );
18
19
20 -- INCIDENTS TABLE
21
22 CREATE TABLE incidents (
23     incident_id SERIAL PRIMARY KEY,
24     type VARCHAR(50) NOT NULL,
25     severity INT CHECK (severity BETWEEN 1 AND 5),
26     reported_time TIMESTAMP NOT NULL,
27     description TEXT,
28     location GEOMETRY(Point, 4326)
29 );
30
31
32 -- HISTORICAL TRAFFIC TABLE
33
34 CREATE TABLE historical_traffic (
35     record_id SERIAL PRIMARY KEY,
36     sensor_id INT REFERENCES sensors(sensor_id),
37     date DATE NOT NULL,
38     avg_speed FLOAT,
```

The status bar at the bottom right shows "Activate Windows Go to Settings to activate Windows.", "Ln 106, Col 25", "Spaces: 4", "UTF-8", "CRLF", "Port: 5500", "30°C Sunny", "ENG IN", "4:13 PM", and "25-Nov-25".

The screenshot shows a code editor window with the title bar "flutter\_windows\_3.35.7-stable". The left sidebar displays a file tree for a Flutter project named "FLUTTER\_WINDOWS\_3.35.7-STABLE". The main editor area contains the following PostgreSQL SQL code:

```
91 -- 4. Find roads affected by incidents (50m distance)
92 SELECT h.record_id, h.road_segment
93 FROM historical_traffic h
94 JOIN incidents i ON ST_DWithin(i.location, h.road_segment, 50);
95
96 -- 5. Count incidents by severity
97 SELECT severity, COUNT(*)
98 FROM incidents
99 GROUP BY severity
100 ORDER BY severity;
101
102
103 -- 6. List all active sensors
104 SELECT sensor_id, sensor_name
105 FROM sensors
106 WHERE status = 'ACTIVE';
```

The status bar at the bottom right shows "Activate Windows Go to Settings to activate Windows.", "Ln 106, Col 25", "Spaces: 4", "UTF-8", "CRLF", "Port: 5500", "30°C Sunny", "ENG IN", "4:14 PM", and "25-Nov-25".



The screenshot shows a Visual Studio Code (VS Code) interface with the title bar "File Edit Selection View Go Run Terminal Help" and the tab "flutter\_windows\_3.35.7-stable". The left sidebar displays a file tree for a project named "FLUTTER\_WINDOWS\_3.35.7-STABLE". The main editor area contains a SQL script for creating a database schema. The script includes a "HISTORICAL\_TRAFFIC\_TABLE" with columns for record\_id, sensor\_id, date, avg\_speed, vehicle\_count, and road\_segment (a GEOMETRY type). It also creates two spatial indexes: "idx\_sensors\_location" on the sensors table and "idx\_incidents\_location" on the incidents table. Additionally, it creates non-spatial indexes: "idx\_incidents\_severity" on the incidents table and "idx\_sensors\_status" on the sensors table. The script concludes with a section titled "TASK 2: EXAMPLE SPATIAL + NORMAL QUERIES". A status bar at the bottom shows "Ln 106 Col 25 Spaces: 4 UTF-8 CR LF { MS SQL } Port: 5500" and a system tray with icons for battery, signal, and time.

```
print("... Irregularity: Missing Check-in")  
# Mapper: assign each event to a sliding window  
-- Enable PostGIS extension  
-- Untitled-7  
  
EXPLORER ... 3-3 ● QR print("... Irregularity: Missing Check-in") Untitled-4 ● # Mapper: assign each event to a sliding window Untitled-5 ● ■ -- Enable PostGIS extension Untitled-6 ● ■ Untitled-7 ● ...  
  
FLUTTER_WINDOWS_3.35.7-STABLE ...  
  vscode  
  auth.js  
  db.js  
  Faculty.js  
  from datetime import datetime.rb  
  Java.js  
  js-project.js  
  lab test wtmp.html  
  launch.json  
  package.json  
  projects.json  
  routes  
  server.ts  
  Untitled-2.md  
  Untitled-2.py  
  flutter  
    .dart_tool  
    .github  
    .idea  
    .pub-preload-cache  
    .vscode  
  bin  
  buildtools  
  dev  
  docs  
  engine  
    scripts  
    src  
      build  
  OUTLINE  
  TIMELINE  
  METADATA  
  
31 -- HISTORICAL_TRAFFIC_TABLE  
32  
33  
34 CREATE TABLE historical_traffic (  
35   record_id SERIAL PRIMARY KEY,  
36   sensor_id INT REFERENCES sensors(sensor_id),  
37   date DATE NOT NULL,  
38   avg_speed FLOAT,  
39   vehicle_count INT,  
40   road_segment GEOMETRY(LineString, 4326)  
41 );  
42  
43 -- TASK 2: CREATE INDEXES (SPATIAL + NORMAL)  
44  
45  
46 -- Spatial indexes  
47 CREATE INDEX idx_sensors_location  
48 | ON sensors USING GIST(location);  
49  
50 CREATE INDEX idx_incidents_location  
51 | ON incidents USING GIST(location);  
52  
53 CREATE INDEX idx_traffic_road_segment  
54 | ON historical_traffic USING GIST(road_segment);  
55  
56 -- Non-spatial indexes  
57 CREATE INDEX idx_incidents_severity  
58 | ON incidents(severity);  
59  
60 CREATE INDEX idx_sensors_status  
61 | ON sensors(status);  
62  
63 CREATE INDEX idx_traffic_date  
64 | ON historical_traffic(date);  
65  
66  
67 -- TASK 2: EXAMPLE SPATIAL + NORMAL QUERIES  
68
```

The screenshot shows a terminal window with the following details:

- Title Bar:** flutter\_windows\_3.35.7-stable
- File Menu:** File, Edit, Selection, View, Go, Run, Terminal, Help
- Terminal Content:**

```
print("--- Irregularity: Missing Check-i Untitled-4")
# Mapper: assign each event to a sliding Untitled-5
-- Enable PostGIS extension Untitled-6
Untitled-7

FLUTTER_WINDOWS_3.35.7-STABLE
...
auth.js
db.js
Faculty.js
from datetime import datetime
Java.js
js-project.js
lab test wtmp.html
launch.json
package.json
projects.json
routes
server.ts
Untitled-2.md
Untitled-2.py
flutter
.dart_tool
.github
.idea
.pub-preload-cache
.vscode
bin
buildtools
dev
docs
engine
scripts
src
build

OUTLINE
TIMELINE
METADATA

Ln 106, Col 25 Spaces: 4 UTF-8 CRLF { MS SQL Port: 5500 30°C Sunny 4:14 PM
Search
4 14 Nov 25
```

## *OUTPUT:*

## **EXPLANATION:**

*This script first enables PostGIS, which adds support for geospatial data in PostgreSQL. Then it creates three tables: sensors (stores sensor details and their GPS point locations), incidents (records traffic incidents with severity, time, and map point), and historical\_traffic (stores daily traffic data with a LINESTRING road segment and reference to a sensor). After creating the tables, the script adds spatial indexes using GIST to speed up map-based queries, and normal indexes on severity, status, and dates to improve filtering performance. Finally, it includes example queries such as finding incidents near a sensor, selecting sensors inside a bounding map area, retrieving traffic history, finding roads near incidents, counting incidents by severity, and listing active sensors. These demonstrate how geospatial and normal SQL queries work in the designed database.*

## **QUE-2:**

### **PROMPT:**

*Write SQL and MapReduce code to detect hotspots in time-series data using sliding windows. Hotspots are defined as regions where the count of events exceeds a threshold within a window. Provide sample queries and validation steps with test datasets.*

## **TASK-1 CODE:**

```
Default x + v

SQL*Plus: Release 11.2.0.2.0 Production on Tue Nov 25 15:38:18 2025
Copyright (c) 1982, 2014, Oracle. All rights reserved.

SQL> connect
Enter user-name: SYSTEM
Enter password:
Connected.
SQL> -- Table: events(event_id, location, event_time)
SQL> -- Detect hotspots in 10-minute sliding windows
SQL>
SQL> WITH windowed AS (
  2   SELECT
  3     location,
  4     event_time,
  5     COUNT(*) OVER (
  6       PARTITION BY location
  7       ORDER BY event_time
  8       RANGE BETWEEN INTERVAL '10' MINUTE PRECEDING AND CURRENT ROW
  9     ) AS window_count
10   FROM events
11 )
12   SELECT location, event_time, window_count
13   FROM windowed
14 WHERE window_count > 10
15 ORDER BY event_time;
  FROM events
*
ERROR at line 10:
ORA-00942: table or view does not exist

SQL> -- Step 1: Create the events table
SQL> CREATE TABLE events (
  2   event_id    NUMBER PRIMARY KEY,
  3   location    VARCHAR2(50),
  4   event_time  TIMESTAMP
  5 );
Table created.

Activate Windows
Go to Settings to activate Windows.

Windows Taskbar: Search, File Explorer, File History, Task View, Taskbar Icons, Start button, Weather (30°C Sunny), Language (ENG IN), Network (Wi-Fi), Date and Time (4:25 PM, 25-Nov-25)
```

```
Default x + v

Table created.

SQL>
SQL> -- Step 2: Insert sample data
SQL> INSERT INTO events VALUES (1, 'A', TIMESTAMP '2025-11-25 15:00:00');

1 row created.

SQL> INSERT INTO events VALUES (2, 'A', TIMESTAMP '2025-11-25 15:01:00');

1 row created.

SQL> INSERT INTO events VALUES (3, 'A', TIMESTAMP '2025-11-25 15:02:00');

1 row created.

SQL> INSERT INTO events VALUES (4, 'A', TIMESTAMP '2025-11-25 15:03:00');

1 row created.

SQL> INSERT INTO events VALUES (5, 'A', TIMESTAMP '2025-11-25 15:04:00');

1 row created.

SQL> INSERT INTO events VALUES (6, 'A', TIMESTAMP '2025-11-25 15:05:00');

1 row created.

SQL> INSERT INTO events VALUES (7, 'A', TIMESTAMP '2025-11-25 15:06:00');

1 row created.

SQL> INSERT INTO events VALUES (8, 'A', TIMESTAMP '2025-11-25 15:07:00');

1 row created.

SQL> INSERT INTO events VALUES (9, 'A', TIMESTAMP '2025-11-25 15:08:00');

1 row created.

SQL> INSERT INTO events VALUES (10, 'A', TIMESTAMP '2025-11-25 15:09:00');

Activate Windows
Go to Settings to activate Windows.

Windows Taskbar: Search, File Explorer, File History, Task View, Taskbar Icons, Start button, Weather (30°C Sunny), Language (ENG IN), Network (Wi-Fi), Date and Time (4:25 PM, 25-Nov-25)
```

## **OUTPUT:**

LOCATION
EVENT_TIME
WINDOW_COUNT
A
25-NOV-25 03.09.30.000000 PM
11

## TASK-2 CODE:

The screenshot shows the VS Code interface with the following details:

- File Bar:** File, Edit, Selection, View, Go, Run, Terminal, Help.
- Title Bar:** flutter\_windows\_3.35.7-stable
- Explorer:** Shows a project structure for "FLUTTER\_WINDOWS\_3.35.7-STABLE".
- Code Editor:** Displays Python code for a Mapper and Reducer. The Mapper code uses `yield` to assign events to windows. The Reducer code counts events per window and yields results based on a threshold.
- Status Bar:** Activate Windows, Go to Settings to activate Windows.
- Bottom Bar:** Includes icons for search, file operations, and system status (30°C, ENG, IN, 4:30 PM, 25-Nov-25).

## OUTPUT:

The terminal output shows the execution of a Python script:

```
PS C:\Users\DELL\Downloads\flutter_windows_3.35.7-stable> Final output (after reducer):
>> (('locA', 0), ('HOTSPOT', 12))
>> (('locB', 1), ('NORMAL', 2))
```

## EXPLANATION:

**Fix:** The error happened because the table `events` didn't exist. Creating it resolves the issue.

**Sample Data:** I inserted 11 events for location A within 10 minutes, so the query will correctly flag a hotspot.

**Query:** Uses Oracle's analytic window function with a 10-minute sliding window.

