```sql
/* Creare una materialized view MV_RESOCONTO avente funzione di reporting, contenente,
per ogni specializzazione il numero di visite effettuate, il numero di nuovi pazienti
visitati, l'incasso totale relativo al mese in corso, e la matricola del medico che ha
visitato più pazienti. Implementare: i) l'incremental refresh; ii) il full refresh
(o rebuild). Per semplicità, implementare solamente il push relativo all'inserimento.
*/

/*Creazione MV*/
DROP TABLE IF EXISTS MV_RESOCONTO;
CREATE TABLE MV_RESOCONTO (
  Specializzazione VARCHAR(100) NOT NULL,
  Visite INT DEFAULT NULL,
  Pazienti INT DEFAULT NULL,
  IncassoMese INT DEFAULT 0,
  MedicoPiuPaz CHAR(3) DEFAULT NULL,
  PRIMARY KEY(Specializzazione)
);

DELIMITER $$
DROP PROCEDURE IF EXISTS medico_piu_paz $$
CREATE PROCEDURE medico_piu_paz(OUT med_ CHAR(3))
BEGIN

    DECLARE n_visite INTEGER DEFAULT 0;
    DECLARE medico CHAR(3) DEFAULT '';

    DECLARE max_visite INTEGER DEFAULT 0;
    DECLARE medico_max_visite CHAR(3) DEFAULT '';

    -- variabile per contare quanti medici a pari merito
    DECLARE ex_aequo INTEGER DEFAULT -1;

    DECLARE finito INTEGER DEFAULT 0;

    DECLARE visite CURSOR FOR
    SELECT V.Medico, COUNT(*)
    FROM Visita V
    WHERE YEAR(V.Data) = YEAR(CURRENT_DATE)
          AND MONTH(V.Data) = MONTH(CURRENT_DATE)
    GROUP BY V.Medico;

    DECLARE CONTINUE HANDLER FOR NOT FOUND
    SET finito = 1;

    OPEN visite;

    scan: LOOP

        FETCH visite INTO  medico, n_visite;
        IF finito = 1 THEN
            LEAVE scan;
        END IF;

        IF n_visite > max_visite THEN
            SET max_visite = n_visite;
            SET medico_max_visite = medico;
            SET ex_aequo = ex_aequo + 1;
        END IF;

    END LOOP scan;

    CLOSE visite;

    -- se ci sono pari merito restituisco NULL
```

```
    -- altrimenti posso eliminare l'IF e restituire sempre
    -- l'ultimo medico trovato fra i pari merito.
    IF ex_aequo <> 0 THEN
        SET med_ = NULL;
    ELSE
        SET med_ = medico_max_visite;
    END IF;

END $$

-- stessa stored procedure, ma dichiarativa
DROP PROCEDURE IF EXISTS medico_piu_paz2$$
CREATE PROCEDURE medico_piu_paz2(OUT med_ CHAR(3))
BEGIN
    DECLARE medico_target CHAR(3) DEFAULT '';

    WITH visite AS
    (
     SELECT V.Medico, COUNT(*) n_visite
     FROM Visita V
     WHERE YEAR(V.Data) = YEAR(CURRENT_DATE)
           AND MONTH(V.Data) = MONTH(CURRENT_DATE)
     GROUP BY V.Medico
    )
    SELECT V1.Medico INTO medico_target
    FROM visite V1
    WHERE n_visite =
        (
         SELECT MAX(n_visite)
         FROM visite V2
        )
    LIMIT 1; -- prendo il primo, se ho pari merito

    SET med_ = medico_target;

END $$


/*Build (Popolamento)*/
DROP PROCEDURE IF EXISTS build $$
CREATE PROCEDURE build()
BEGIN
    DECLARE medico_top CHAR(3) DEFAULT '';

    CALL medico_piu_paz(medico_top);

    INSERT INTO MV_RESOCONTO
    WITH incasso_mese AS
    (
     SELECT M.Specializzazione,
            SUM(M.Parcella) AS Incasso
     FROM Visita V
          INNER JOIN
          Medico M ON V.Medico = M.Matricola
     WHERE MONTH(V.Data) = MONTH(CURRENT_DATE)
           AND YEAR(V.Data) = YEAR(CURRENT_DATE)
           AND V.Mutuata = 0
     GROUP BY M.Specializzazione
    )
    SELECT M.Specializzazione,
           COUNT(*) AS Visite,
           COUNT(DISTINCT V.Paziente) AS Pazienti,
           IM.Incasso AS IncassoMeseCorrente,
           medico_top
```

```
    FROM Visita V
         INNER JOIN
         Medico M ON V.Medico = M.Matricola
         NATURAL JOIN
         incasso_mese IM
    GROUP BY M.Specializzazione, IM.Incasso;
END $$

/* Full refresh */
DROP PROCEDURE IF EXISTS full_refresh $$
CREATE PROCEDURE full_refresh()
BEGIN
    TRUNCATE MV_RESOCONTO;
    CALL build();
END $$

/* Incremental refresh */

/*Creazione log table*/
CREATE TABLE RESOCONTO_LOG(
    Istante TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
    Medico CHAR(50) NOT NULL,
    Paziente CHAR(50) NOT NULL,
    NuovoPaz TINYINT(1) NOT NULL, -- TINYINT(1) è un booleano
    PRIMARY KEY (Istante)
)$$

DROP TRIGGER IF EXISTS push $$

CREATE TRIGGER push
AFTER INSERT ON Visita
FOR EACH ROW
BEGIN
    DECLARE visite_prec INTEGER DEFAULT 0;

    SELECT COUNT(*) - 1
    FROM Visita V
    WHERE V.Medico = NEW.Medico
          AND V.Paziente = NEW.Paziente
    INTO visite_prec;

    INSERT INTO RESOCONTO_LOG(Medico,Paziente,NuovoPaz)
    VALUES(NEW.Paziente, NEW.Medico, IF(visite_prec>0,0,1));
END $$


DROP PROCEDURE IF EXISTS refresh $$
CREATE PROCEDURE refresh(policy CHAR(20), _up_to TIMESTAMP)
BEGIN
    DECLARE medico_top CHAR(3) DEFAULT '';

    IF policy = 'rebuild' THEN

        CALL full_refresh();

    ELSEIF (policy = 'partial' OR policy = 'complete') THEN

        CALL medico_piu_paz(medico_top);

        REPLACE INTO MV_Resoconto
        WITH log_processed AS (
            SELECT M.Specializzazione,
                   COUNT(*) AS NuoveVisite,
                   SUM(NuovoPaz) AS NuoviPazienti,
```

```
                    SUM(M.Parcella) AS NuovoIncasso
            FROM RESOCONTO_LOG RL
                 INNER JOIN
                 Medico M ON RL.Medico = M.Matricola
            WHERE RL.Istante <=
                IF(policy = 'complete',
                   CURRENT_TIMESTAMP,
                   _up_to)
            GROUP BY M.Specializzazione
        )
        SELECT M.Specializzazione,
               IF(MV.Specializzazione IS NULL, LP.NuoveVisite, MV.Visite+LP.NuoveVisite),
               IF(MV.Specializzazione IS NULL, LP.NuoviPazienti,
MV.Pazienti+LP.NuoviPazienti),
               IF(MV.Specializzazione IS NULL, LP.NuovoIncasso,
MV.IncassoMese+LP.NuovoIncasso),
               medico_top
        FROM
            log_processed LP
            -- faccio join esterno perché ci possono essere
            -- record nel log non associati a
            -- specializzazioni per le quali c'è già
            -- un record nella MV!
            LEFT OUTER JOIN
            MV_Resoconto MV USING(Specializzazione)
        GROUP BY LP.Specializzazione;

        IF policy = 'complete' THEN
            DELETE
            FROM RESOCONTO_LOG;
        ELSE
            DELETE
            FROM RESOCONTO_LOG
            WHERE Istante <= _up_to;
        END IF;

    ELSE
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Policy errata!';
    END IF;
END $$

DELIMITER ;
```