Query

-Visualizza in ordine temporale tutti i treni che vanno da un punto a ad un punto b e il loro relativi costi data una classe di viaggio

```
SELECT spostamento.percorso, spostamento.giorno, entrata.entrata_programmata,
uscita.uscita_programmata,
sum((tratta.lunghezza)/1000)*(SELECT prezzo
                              FROM classe_viaggio
                              WHERE classe_viaggio.nome = ClasseViaggio) as costo

FROM spostamento, percorso p0, sezione viaggio, sezione entrata, sezione uscita,
tratta
WHERE spostamento.percorso = p0.identificativo
AND spostamento.percorso = p0.identificativo
AND viaggio.percorso = p0.identificativo
AND viaggio.tratta_fine = tratta.fine
AND viaggio.tratta_inizio = tratta.inizio
AND viaggio.tratta_limit = tratta.limite_velocità
AND (entrata.percorso,entrata.ordine) IN (SELECT s1.percorso, s1.ordine
     FROM percorso p1, sezione s1
     WHERE p1.identificativo = p0.identificativo
     AND s1.percorso = p0.identificativo
     AND s1.tratta_inizio = partenzaA
     AND (s1.ordine = 0 OR (SELECT s2."fermata?"
                            FROM sezione s2
                            WHERE s2.ordine = s1.ordine - 1
                            AND s2.percorso = p0.identificativo)))
AND (uscita.percorso,uscita.ordine) IN (SELECT s3.percorso, s3.ordine
     FROM percorso p3, sezione s3
     WHERE p3.identificativo = p0.identificativo
     AND s3.percorso = p0.identificativo
     AND s3.tratta_fine = arrivoB
     AND s3."fermata?" = true)
AND viaggio.ordine >= entrata.ordine
AND viaggio.ordine <= uscita.ordine
GROUP BY spostamento.percorso, spostamento.giorno, entrata.entrata_programmata,
uscita.uscita_programmata
ORDER BY spostamento.giorno, entrata.entrata_programmata
```

-Visualizza il numero della carrozza con valore intero minore che ha ancora posti disponibili per un dato viaggio

```
SELECT MIN(c.numero) INTO numeroCarrozza
        FROM carrozza c
        WHERE c.classe_viaggio= ClasseViaggio
        AND c.idtreno = IdentificativoTreno
        AND (SELECT count(*)
             FROM posto
             WHERE posto.idtreno = c.idtreno
      AND c.numero = posto.carrozza) != (SELECT count(DISTINCT biglietto.posto)
               FROM biglietto, sezione partenza, sezione arrivo
               WHERE biglietto.spostamento_percorso = IdentificativoPercorso
               AND biglietto.data_viaggio = DataViaggio
               AND biglietto.carrozza = c.numero
               AND partenza.percorso = biglietto.spostamento_percorso
               AND arrivo.percorso = biglietto.spostamento_percorso
               AND partenza.ordine = biglietto.partenza_ordine
               AND arrivo.ordine = biglietto.arrivo_ordine
     AND ((arrivo.ordine >= ordinePartenza AND partenza.ordine <= ordineArrivo)
     OR (partenza.ordine <= ordineArrivo AND arrivo.ordine >= ordinePartenza)))
```

-Visualizza il numero del posto con valore intero minore libero per un dato viaggio

```
SELECT MIN(p.numero) INTO numeroPosto
                from posto p
                where p.numero NOT IN (SELECT DISTINCT biglietto.posto
                FROM biglietto, sezione partenza, sezione arrivo
                WHERE biglietto.spostamento_percorso = IdentificativoPercorso
                AND biglietto.data_viaggio = DataViaggio
                AND biglietto.carrozza = NumeroCarrozza
                AND partenza.percorso = biglietto.spostamento_percorso
                AND arrivo.percorso = biglietto.spostamento_percorso
                AND partenza.ordine = biglietto.partenza_ordine
                AND arrivo.ordine = biglietto.arrivo_ordine
     AND ((arrivo.ordine >= ordinePartenza AND partenza.ordine <= ordineArrivo)
     OR (partenza.ordine <= ordineArrivo AND arrivo.ordine >= ordinePartenza)))
                        AND p.idtreno = IdentificativoTreno
                        AND p.carrozza = NumeroCarrozza;
```

-Generalità dei segretari che hanno risposto ad una assistenza clienti relativa al biglietto più costoso venduto in quella giornata

```
SELECT impiegato.*, persona.*
from persona, impiegato
WHERE persona.email = impiegato.email
AND impiegato.cf IN (SELECT impiegato
         FROM segretario, contenuto_assistenza, assistenza_clienti
         WHERE contenuto_assistenza.segretario = segretario.impiegato
         AND assistenza_clienti.identificativo = contenuto_assistenza.assistenza
         AND assistenza_clienti.biglietto IN (SELECT b1.identificativo
                        FROM biglietto b1
                        WHERE b1.identificativo = assistenza_clienti.biglietto
                        AND b1.prezzo >= ALL (SELECT b2.prezzo
                                        FROM biglietto b2
                                        WHERE b2.data_acquisto = b1.data_acquisto)))
```

Con View

```
CREATE VIEW Biglietti_Costosi AS
    SELECT b1.identificativo, b1.data_acquisto
    FROM biglietto b1
    WHERE b1.prezzo >= ALL (SELECT b2.prezzo
                            FROM biglietto B2
                            WHERE b1.data_acquisto = b2.data_acquisto);

SELECT impiegato.*, persona.*
from persona, impiegato
WHERE persona.email = impiegato.email
AND impiegato.cf IN (SELECT impiegato
         FROM segretario, contenuto_assistenza, assistenza_clienti
         WHERE contenuto_assistenza.segretario = segretario.impiegato
         AND assistenza_clienti.identificativo = contenuto_assistenza.assistenza
         AND assistenza_clienti.biglietto IN (SELECT identificativo FROM
Biglietti_Costosi));
```

-Biglietto più costoso acquistato da ogni account

```
SELECT a1.codice_fedeltà, b1.identificativo, b1.prezzo
FROM account a1, biglietto b1
WHERE b1.account = a1.codice_fedeltà
AND b1.prezzo >= all (SELECT b2.prezzo
                        FROM biglietto b2
                        WHERE b2.account = a1.codice_fedeltà)
```

-Posizione finale del treno nel futuro dopo aver fatto tutti gli spostamenti programmati

```
SELECT p0.arrivo, p0.ora_arrivo, s0.giorno
FROM spostamento s0, percorso p0
WHERE s0.percorso = p0.identificativo
```

```
AND s0.treno = new.treno
AND s0.giorno = (SELECT max(s1.giorno)
                 FROM spostamento s1
                 WHERE s1.treno = new.treno)
AND p0.ora_arrivo >= all (SELECT ora_partenza
                          FROM percorso p1, spostamento s1
                          WHERE s1.percorso = p1.identificativo
                          AND s1.treno = new.treno
                          AND s1.giorno = (SELECT MAX(s2.giorno)
                                           FROM spostamento s2
                                           WHERE s2.treno = new.treno));
```

- Treni in partenza in una stazione, il loro orario di partenza e la loro destinazione

```
SELECT t1.identificativo, p1.identificativo, p1.arrivo,  s1.giorno,
sez1.entrata_programmata, binario.binario
FROM treno t1, percorso p1, spostamento s1, sezione sez1, sezione binario
WHERE s1.treno = t1.identificativo
AND s1.percorso = p1.identificativo
AND sez1.percorso = p1.identificativo
AND sez1.tratta_inizio = 'Pescara'
AND (sez1.ordine = 0 OR (SELECT "fermata?"
  FROM sezione
  WHERE sezione.percorso = sez1.percorso
  AND sezione.ordine = sez1.ordine - 1))
--Comando per trovare il binario (visto che si riferisce alla sezione
precendente (che potrebbe anche essere in un altro
--spostamento del tutto)
AND (binario.ordine, binario.percorso) IN (SELECT sez2.ordine, sez2.percorso
              FROM sezione sez2, percorso p2, spostamento s2
              WHERE
                 (   sez2.percorso = sez1.percorso
                 AND sez1.ordine > 0
                 AND sez2.tratta_fine = 'Pescara')
              OR
                 (   sez1.ordine = 0
                 AND p2.identificativo = (SELECT p3.identificativo
                         FROM percorso p3, spostamento s3
                         WHERE s3.treno = t1.identificativo
                         AND p3.identificativo = s3.percorso
                         AND p3.ora_arrivo < p1.ora_partenza
                         AND p3.ora_arrivo >= ALL (SELECT p4.ora_arrivo
                                                   FROM percorso p4, spostamento s4
                                                   WHERE p4.identificativo = s4.percorso
                                                   AND s4.treno = t1.identificativo
                                                   AND p4.ora_arrivo < p1.ora_partenza))
                                    AND sez2.ordine = (SELECT max(sez3.ordine)
                                    FROM sezione sez3
                                    WHERE sez3.percorso = sez2.percorso))))
AND (s1.giorno > current_date OR (s1.giorno = current_date AND
sez1.entrata_programmata > current_time))
ORDER BY s1.giorno, sez1.entrata_programmata ASC;
```

-Treni in arrivo in una stazione, il loro orario di arrivo e la loro provenienza

```
SELECT t1.identificativo, p1.identificativo, p1.partenza, s1.giorno,
sez1.uscita_programmata, sez1.binario
    FROM treno t1, percorso p1, spostamento s1, sezione sez1
    WHERE s1.treno = t1.identificativo
    AND s1.percorso = p1.identificativo
    AND sez1.percorso = p1.identificativo
    AND sez1.tratta_fine = stazione
    AND sez1."fermata?"
    AND (s1.giorno > current_date OR (s1.giorno = current_date AND
```

```
sez1.uscita_programmata > current_time))
   ORDER BY s1.giorno, uscita_programmata ASC;
```

-Treni che arrivano a Roma provenienti da Milano che hanno più di 300 passeggeri

```
SELECT treno.identificativo, treno.modello_treno, spostamento.percorso,
spostamento.giorno
FROM treno, spostamento, percorso
WHERE spostamento.treno = treno.identificativo
AND spostamento.percorso = percorso.identificativo
AND (SELECT count(*)
     FROM biglietto
     WHERE biglietto.data_viaggio = spostamento.giorno
     AND biglietto.spostamento_percorso = spostamento.percorso) > 300
AND (spostamento.percorso, spostamento.giorno) IN (SELECT s1.percorso, s1.giorno
                                                   FROM spostamento s1, percorso
p1
                                                   WHERE s1.percorso =
p1.identificativo
                                                   AND p1.partenza = 'Milano'
                                                   AND p1.arrivo = 'Roma')
```

Con view

```
CREATE VIEW Spostamenti_Popolati AS
    SELECT biglietto.spostamento_percorso, biglietto.data_viaggio
    FROM biglietto
    GROUP BY biglietto.spostamento_percorso
    HAVING count(*) > 300;

SELECT treno.identificativo, treno.modello_treno, spostamento.percorso,
spostamento.giorno
FROM treno, spostamento, percorso, Spostamenti_Popolati
WHERE spostamento.treno = treno.identificativo
AND spostamento.percorso = percorso.identificativo
AND spostamento.giorno = Spostamenti_Popolati.data_viaggio
AND spostamento.percorso = Spostamenti_Popolati.spostamento_percorso
AND (spostamento.percorso, spostamento.giorno) IN (SELECT s1.percorso, s1.giorno
                                                   FROM spostamento s1, percorso p1
                                                   WHERE s1.percorso = p1.identificativo
                                                   AND p1.partenza = 'Milano'
                                                   AND p1.arrivo = 'Roma')
```

-Dati degli impiegati che lavorano in un dato spostamento

```
SELECT *
FROM impiegato
WHERE cf IN (SELECT ferroviere.impiegato
             FROM ferroviere, spostamento, personale_bordo
             WHERE 'percorso'= spostamento.percorso
             AND 'giorno' = spostamento.giorno
             AND spostamento.idgruppo = personale_bordo.identificativo
             AND (capotreno = ferroviere.impiegato OR macchinista =
ferroviere.impiegato))
OR cf IN     (SELECT assistente.impiegato
             FROM assistente, spostamento, servizio_assistente
             WHERE 'percorso'= spostamento.percorso
             AND 'giorno' = spostamento.giorno
             AND spostamento.idgruppo = servizio_assistente.idgruppo
             AND servizio_assistente.assistente = assistente.impiegato)
```

-Totale da pagare a partire da un giorno a scelta fino a fine mese per un impiegato (fino alla fine del contratto attuale nel caso finisca nel mezzo del mese)

```
SELECT SUM(extract(HOUR from (t.fine-t.inizio))) * (SELECT c.paga_oraria
                                    FROM contratto c
```

```
                                    WHERE CodiceFiscale=c.impiegato
                                    AND c.rescisso = false
                                    AND c.fine >= Inizio_Calcolo
                                    AND c.inizio <= Inizio_Calcolo) INTO somma
    from turno t, contratto c
    where extract(MONTH FROM t.giorno) = extract(MONTH FROM Inizio_Calcolo)
    AND t.impiegato = c.impiegato
    AND t.giorno <= InizioCalcolo
    AND t.giorno >= c.inizio
    AND t.giorno <= c.fine
    AND t.impiegato = CodiceFiscale;
```

Con view (leggermente diversa, invece di specificare un giorno a scelta, sceglie il giorno attuale. Utile nel caso si voglia eseguire il calcolo automaticamente ogni ultimo del mese)

```
CREATE VIEW paga_attuale AS
SELECT contratto.impiegato, contratto.paga_oraria
FROM contratto
WHERE rescisso = false
AND contratto.fine >= current_date
AND contratto.inizio <= current_date;

SELECT SUM(extract(HOUR from (t.fine-t.inizio))) * (SELECT p.paga_oraria
                                    FROM paga_attuale p
                                    WHERE p.impiegato = CodiceFiscale) INTO
somma
    from turno t, contratto c
    where extract(MONTH FROM t.giorno) = extract(MONTH FROM current_date)
    AND t.impiegato = c.impiegato
    AND t.giorno <= current_date
    AND t.giorno >= c.inizio
    AND t.giorno <= c.fine
    AND t.impiegato = CodiceFiscale;
```

-Lista degli impiegati con contratto valido che non hanno un turno impostato nel futuro

```
SELECT impiegato.*
FROM impiegato
WHERE cf in (SELECT contratto.impiegato
             FROM contratto
             WHERE contratto.inizio < current_date
             AND contratto.fine > current_date)
AND cf not in (SELECT turno.impiegato
           FROM turno
           WHERE turno.giorno > current_date)
```

Con view

```
CREATE VIEW impiegato_corrente AS
    SELECT impiegato.*
    FROM impiegato
    WHERE cf in (SELECT contratto.impiegato
             FROM contratto
             WHERE contratto.inizio < current_date
             AND contratto.fine > current_date);

SELECT impiegato_corrente.*
FROM impiegato_corrente
WHERE impiegato_corrente.cf not in (SELECT turno.impiegato
                                    FROM turno
                                    WHERE turno.giorno > current_date)
```

-Dati personali e lavorativi delle persone che posseggono un account di livello Platino che sono anche operai che hanno effettuato più di 10 manutenzioni

```
SELECT persona.*, impiegato.*
FROM persona, account, impiegato, operaio o
WHERE persona.email = impiegato.email
AND o.impiegato = impiegato.cf
AND account.email = persona.email
AND account.livello = 'Platinum'
AND (SELECT count(*)
     FROM riparatori
     WHERE riparatori.operaio = o.impiegato) > 10;
```

Con view

```
CREATE VIEW persona_impiegato as
SELECT persona.*, impiegato.*
FROM persona, impiegato
WHERE persona.email = impiegato.email;

CREATE VIEW account_platino AS
SELECT account.email
FROM account
WHERE account.livello = 'Platinum';

SELECT persona_impiegato.*
FROM persona_impiegato, operaio o, account_platino
WHERE persona_impiegato.email = account_platino.email
AND o.impiegato = persona_impiegato.cf
AND (SELECT count(*)
     FROM riparatori
     WHERE riparatori.operaio = o.impiegato) > 10;
```

- La manutenzione, il treno e il numero di operai alla quale hanno lavorato più operai per ogni treno che ha avuto almeno una manutenzione

```
SELECT m1.identificativo, m1.idtreno, count(riparatori.operaio)
FROM riparatori, manutenzione m1
WHERE riparatori.manutenzione = m1.identificativo
GROUP BY riparatori.manutenzione, m1.idtreno
HAVING count(riparatori.operaio) >= ALL (SELECT count(r2.operaio)
                                         FROM riparatori r2, manutenzione m2
                                         WHERE r2.manutenzione = m2.identificativo
                                         AND m2.idtreno = m1.idtreno);
```

Con view

```
CREATE VIEW numero_operai_manutenzione AS
    SELECT count(riparatori.operaio) as Numero_Operai, m.identificativo
    FROM riparatori, manutenzione m
    WHERE riparatori.manutenzione = m.identificativo
    GROUP BY m.identificativo;

SELECT manutenzione.identificativo, manutenzione.idtreno,
numero_operai_manutenzione.Numero_Operai
FROM numero_operai_manutenzione, manutenzione
WHERE numero_operai_manutenzione.identificativo = manutenzione.identificativo
AND numero_operai_manutenzione.Numero_Operai >= all (SELECT n2.Numero_Operai
                                FROM numero_operai_manutenzione n2, manutenzione m2
                                WHERE n2.identificativo = m2.identificativo
                                AND m2.idtreno = manutenzione.idtreno)
```

-Coppie di capotreni che hanno un contratto valido e hanno effettuato gli stessi percorsi

```
SELECT f1.impiegato, f2.impiegato
FROM ferroviere f1, ferroviere f2, contratto c1, contratto c2
WHERE f1.impiegato = c1.impiegato
AND c1.inizio > current_date
AND c1.fine < current_date
```

```
AND NOT c1.rescisso
AND f1.ruolo = 'Capotreno'
AND f2.impiegato = c2.impiegato
AND c2.inizio > current_date
AND c2.fine < current_date
AND NOT c2.rescisso
AND f2.ruolo = 'Capotreno'
AND NOT EXISTS(
    (SELECT percorso.identificativo
     FROM percorso, spostamento, personale_bordo
     WHERE spostamento.percorso = percorso.identificativo
     AND spostamento.idgruppo = personale_bordo.identificativo
     AND capotreno = f1.impiegato) EXCEPT (SELECT percorso.identificativo
                        FROM percorso, spostamento, personale_bordo
                        WHERE spostamento.percorso = percorso.identificativo
                        AND spostamento.idgruppo = personale_bordo.identificativo
                        AND capotreno = f2.impiegato))
AND NOT EXISTS(
    (SELECT percorso.identificativo
     FROM percorso, spostamento, personale_bordo
     WHERE spostamento.percorso = percorso.identificativo
     AND spostamento.idgruppo = personale_bordo.identificativo
     AND capotreno = f2.impiegato) EXCEPT (SELECT percorso.identificativo
                        FROM percorso, spostamento, personale_bordo
                        WHERE spostamento.percorso = percorso.identificativo
                        AND spostamento.idgruppo = personale_bordo.identificativo
                        AND capotreno = f1.impiegato));
```

Con view

```
SELECT f1.impiegato, f2.impiegato
FROM capotreni_attuali f1, capotreni_attuali f2
WHERE NOT EXISTS(
    (SELECT percorso.identificativo
     FROM percorso, spostamento, personale_bordo
     WHERE spostamento.percorso = percorso.identificativo
     AND spostamento.idgruppo = personale_bordo.identificativo
     AND capotreno = f1.impiegato) EXCEPT (SELECT percorso.identificativo
                        FROM percorso, spostamento, personale_bordo
                        WHERE spostamento.percorso = percorso.identificativo
                        AND spostamento.idgruppo = personale_bordo.identificativo
                        AND capotreno = f2.impiegato))
AND NOT EXISTS(
    (SELECT percorso.identificativo
     FROM percorso, spostamento, personale_bordo
     WHERE spostamento.percorso = percorso.identificativo
     AND spostamento.idgruppo = personale_bordo.identificativo
     AND capotreno = f2.impiegato) EXCEPT (SELECT percorso.identificativo
                        FROM percorso, spostamento, personale_bordo
                        WHERE spostamento.percorso = percorso.identificativo
                        AND spostamento.idgruppo = personale_bordo.identificativo
                        AND capotreno = f1.impiegato))
```

Operazioni

Inserimento Persona

```
INSERT INTO persona (nome, cognome, email, cellulare) VALUES ('...', '...',
'...', '...');
```

Creazione di un account

```
INSERT INTO account (codice_fedeltà, account_password, email)
VALUES ('...', '...', '...', '...');
```

Acquisto Biglietto

```
CREATE FUNCTION acquisto_biglietto(emailnew email, spostgiornonew date,
spostpercorsonew integer, classeviaggionew character varying, partenzaf
character varying, arrivof character varying)

RETURNS TABLE("IdBiglietto" integer, "Nome" character varying, "Cognome"
character varying, "Treno" smallint, "Carrozza" smallint, "Posto" smallint,
"Partenza" character varying, "Ora Partenza" time without time zone, "Arrivo"
character varying, "Ora Arrivo" time without time zone, "Data_Viaggio" date,
"Prezzo Biglietto" numeric)
    LANGUAGE plpgsql
AS
$$
DECLARE
        numeroPosto integer;
        numeroCarrozza integer;
        ordinePartenza integer;
        ordineArrivo integer;
        costoKM integer;
        metriTot integer;
        accountUt integer;
        trenoSpost integer;
        valore INTEGER;
    BEGIN
        SELECT spos.treno INTO trenoSpost
        FROM spostamento spos
        WHERE spos.giorno=spostGiornoNew AND spos.percorso=spostPercorsoNew;

        SELECT ordine INTO ordinePartenza
        FROM sezione
        WHERE sezione.percorso = spostpercorsonew
        AND sezione.tratta_inizio = partenzaf;

        SELECT ordine INTO ordineArrivo
        FROM sezione
        WHERE sezione.percorso = spostpercorsonew
        AND sezione.tratta_fine = arrivof;

        SELECT MIN(c.numero) INTO numeroCarrozza
        FROM carrozza c
        WHERE c.classe_viaggio=classeViaggioNew
        AND c.idtreno = trenoSpost
        AND (SELECT count(*)
             FROM posto
             WHERE posto.idtreno = c.idtreno
             AND c.numero = posto.carrozza) != (SELECT count(DISTINCT
biglietto.posto)
                        FROM biglietto, sezione partenza, sezione arrivo
                        WHERE biglietto.spostamento_percorso = spostPercorsoNew
                        AND biglietto.data_viaggio = spostgiornonew
                        AND biglietto.carrozza = c.numero
```

```
                        AND partenza.percorso = biglietto.spostamento_percorso
                        AND arrivo.percorso = biglietto.spostamento_percorso
                        AND partenza.ordine = biglietto.partenza_ordine
                        AND arrivo.ordine = biglietto.arrivo_ordine
                        AND ((arrivo.ordine >= ordinePartenza AND
partenza.ordine <= ordineArrivo)
                        OR (partenza.ordine <= ordineArrivo AND arrivo.ordine >=
ordinePartenza)));
        IF numeroCarrozza IS NULL THEN
            RAISE EXCEPTION 'Non sono rimasti posti';
        END IF;

        SELECT MIN(p.numero) INTO numeroPosto
                    from posto p
                    where p.numero NOT IN (SELECT DISTINCT biglietto.posto
                        FROM biglietto, sezione partenza, sezione arrivo
                        WHERE biglietto.spostamento_percorso = spostPercorsoNew
                        AND biglietto.data_viaggio = spostgiornonew
                        AND biglietto.carrozza = numeroCarrozza
                        AND partenza.percorso = biglietto.spostamento_percorso
                        AND arrivo.percorso = biglietto.spostamento_percorso
                        AND partenza.ordine = biglietto.partenza_ordine
                        AND arrivo.ordine = biglietto.arrivo_ordine
                        AND ((arrivo.ordine >= ordinePartenza AND
partenza.ordine <= ordineArrivo)
                        OR (partenza.ordine <= ordineArrivo AND arrivo.ordine >=
ordinePartenza)))
                        AND p.idtreno = trenoSpost
                        AND p.carrozza = numeroCarrozza;

        IF numeroPosto IS NULL THEN
            RAISE EXCEPTION 'Cè stato un problema';
        END IF;


        SELECT ac.codice_fedeltà INTO accountUt
        FROM account ac
        WHERE emailNew = ac.email;

        SELECT prezzo INTO costoKM
        FROM classe_viaggio
        WHERE classeViaggioNew = classe_viaggio.nome;

        SELECT sum(tratta.lunghezza) INTO metriTot
                FROM tratta, sezione
                WHERE tratta.fine = sezione.tratta_fine
                AND tratta.inizio = sezione.tratta_inizio
                AND tratta.limite_velocità = sezione.tratta_limit
                AND sezione.percorso = spostPercorsoNew
                AND sezione.ordine >= (SELECT ordine
                                       FROM sezione s1
                                       WHERE s1.percorso = spostPercorsoNew
                                       AND s1.tratta_inizio = partenzaF)
                AND sezione.ordine <= (SELECT ordine
                                       FROM sezione s1
                                       WHERE s1.percorso = spostPercorsoNew
                                       AND s1.tratta_fine = arrivoF);

        IF metriTot = 0 OR metriTot IS NULL THEN
            RAISE EXCEPTION 'Errore sul calcolo della lunghezza della tratta';
        END IF;

        INSERT INTO biglietto(prezzo, rimborsato, metodopagamento, id_pagamento,
```

```
account, spostamento_percorso, posto, carrozza, email, treno, data_acquisto,
partenza_ordine, arrivo_ordine, data_viaggio)
        VALUES ((metriTot/1000)*costoKM, false, 'Mastercard', '...', accountUt,
spostPercorsoNew, numeroPosto, numeroCarrozza, emailNew, trenoSpost,
current_date, ordinePartenza, ordineArrivo, spostgiornonew)
        RETURNING identificativo INTO valore;

        RETURN QUERY
        SELECT biglietto.identificativo, persona.nome, persona.cognome,
biglietto.treno, biglietto.carrozza, biglietto.posto, percorso.partenza,
percorso.ora_partenza, percorso.arrivo, percorso.ora_arrivo,
biglietto.data_viaggio, biglietto.prezzo
        FROM biglietto, percorso, persona
        WHERE biglietto.identificativo = valore
        AND percorso.identificativo = spostpercorsonew
        AND persona.email = emailnew;
    end;
$$;
```

Visualizzazione Viaggi

```
CREATE FUNCTION cerca_treno(partenzaf character varying, arrivof character
varying, classe character varying)
    RETURNS TABLE(percorso integer, giorno date, ora_partenza time without time
zone, ora_arrivo time without time zone, costo numeric)
    LANGUAGE plpgsql
AS
$$
DECLARE
    BEGIN
    RETURN QUERY
    SELECT spostamento.percorso, spostamento.giorno,
entrata.entrata_programmata, uscita.uscita_programmata,
(sum(tratta.lunghezza)/1000)*(SELECT prezzo

FROM classe_viaggio

WHERE classe_viaggio.nome = classe) as costo
    FROM spostamento, percorso p0, sezione viaggio, sezione entrata, sezione
uscita, tratta
    WHERE spostamento.percorso = p0.identificativo
    AND spostamento.percorso = p0.identificativo
    AND viaggio.percorso = p0.identificativo
    AND viaggio.tratta_fine = tratta.fine
    AND viaggio.tratta_inizio = tratta.inizio
    AND viaggio.tratta_limit = tratta.limite_velocità
    AND (entrata.percorso,entrata.ordine) IN (SELECT s1.percorso, s1.ordine
         FROM percorso p1, sezione s1
         WHERE p1.identificativo = p0.identificativo
         AND s1.percorso = p0.identificativo
         AND s1.tratta_inizio = partenzaf
         AND (s1.ordine = 0 OR (SELECT s2."fermata?"
                                FROM sezione s2
                                WHERE s2.ordine = s1.ordine - 1
                                AND s2.percorso = p0.identificativo)))
    AND (uscita.percorso,uscita.ordine) IN (SELECT s3.percorso, s3.ordine
         FROM percorso p3, sezione s3
         WHERE p3.identificativo = p0.identificativo
         AND s3.percorso = p0.identificativo
         AND s3.tratta_fine = arrivof
         AND s3."fermata?" = true)
    AND viaggio.ordine >= entrata.ordine
    AND viaggio.ordine <= uscita.ordine
```

```
    GROUP BY spostamento.percorso, spostamento.giorno,
entrata.entrata_programmata, uscita.uscita_programmata;
    ORDER BY spostamento.giorno, entrata.entrata_programmata;
END;
$$;
```

Inserimento Percorso

```
INSERT INTO percorso (identificativo, ora_arrivo, ora_partenza, partenza,
arrivo, tipologia)
VALUES ('...', '...', '...', '...', '...', '...');
```

Inserimento Spostamento

```
INSERT INTO spostamento (percorso, treno, giorno, idgruppo)
VALUES ('...', '...', '...', '...');
```

Nuovo Contratto

```
INSERT INTO contratto (identificativo, tipo, inizio, rescisso, paga_oraria,
fine)
VALUES ('...', '...', '...', '...', '...', '...');
```

Inserimento Nuovo Impiegato

```
INSERT INTO impiegato (cf, email, email_aziendale, data_nascita, iban, sesso,
tipologia)
VALUES ('...', '...', '...', '...', '...', '...', '...');
```

Calcolo Ritardo medio

```
CREATE FUNCTION calcolo_ritardo(giornata character varying)
    RETURNS TABLE(avg interval)
    LANGUAGE plpgsql
AS
$$
BEGIN
        return QUERY
        SELECT avg(sez_eff.uscita - sez.uscita_programmata)
        FROM sezione sez, sezione_effettiva sez_eff
        WHERE sez_eff.giorno = giornata::DATE AND (sez.percorso=sez_eff.percorso
AND sez.ordine = sez_eff.sezione_ordine);
    end;
$$;
```

Calcolo Guadagno

```
create function guadagno_giornata() returns bigint
    language plpgsql
as
$$
DECLARE
        somma int8;
    BEGIN
        SELECT SUM(prezzo) INTO somma
        from biglietto
        where biglietto.data_acquisto::date = current_date AND NOT
biglietto.rimborsato;

        RETURN somma;
    end;
$$;
```

Inserimento Manutenzione

```
INSERT INTO manutenzione (descrizione, idtreno, fine, inizio, banco,
descrizione_guasto)
VALUES ('...', '...', '...', '...', '...', '...');
```

Inserimento Personale di Bordo

```
create function inserimento_personale_bordo(capotrenonew codice_fiscale,
macchinistanew codice_fiscale, assistenti codice_fiscale[]) returns void
    language plpgsql
as
$$
DECLARE
    valore integer;
    cf codice_fiscale;
BEGIN
        INSERT INTO personale_bordo (capotreno, macchinista) VALUES
(capotrenoNew, macchinistaNew);
        SELECT MAX(identificativo) into valore from personale_bordo;
        foreach cf in array assistenti
            LOOP
            INSERT INTO servizio_assistente (assistente, idgruppo) values (cf,
valore);
            end loop;
    END
$$;
```

Visualizza Arrivi in Stazione

```
CREATE FUNCTION visualizza_arrivi(stazione character varying)
    RETURNS TABLE("ID Treno" smallint, "ID Percorso" integer, "Provenienza"
character varying, "Giorno Arrivo" date, "Ora Arrivo" time without time zone)
    LANGUAGE plpgsql
AS
$$
BEGIN
    RETURN QUERY
    SELECT t1.identificativo, p1.identificativo, p1.partenza, s1.giorno,
sez1.uscita_programmata, sez1.binario
    FROM treno t1, percorso p1, spostamento s1, sezione sez1
    WHERE s1.treno = t1.identificativo
    AND s1.percorso = p1.identificativo
    AND sez1.percorso = p1.identificativo
    AND sez1.tratta_fine = stazione
    AND sez1."fermata?"
    AND (s1.giorno > current_date OR (s1.giorno = current_date AND
sez1.uscita_programmata > current_time))
   ORDER BY s1.giorno, uscita_programmata ASC;
END

$$;
```

Visualizza Partenze in Stazione

```
CREATE FUNCTION visualizza_partenze(stazione character varying)
    RETURNS TABLE("ID Treno" smallint, "ID Percorso" integer, "Destinazione"
character varying, "Giorno Partenza" date, "Ora Partenza" time without time
zone)
    LANGUAGE plpgsql
AS
$$
BEGIN
    RETURN QUERY
  SELECT t1.identificativo, p1.identificativo, p1.arrivo,  s1.giorno,
sez1.entrata_programmata, binario.binario
FROM treno t1, percorso p1, spostamento s1, sezione sez1, sezione binario
WHERE s1.treno = t1.identificativo
AND s1.percorso = p1.identificativo
AND sez1.percorso = p1.identificativo
AND sez1.tratta_inizio = stazione
AND (sez1.ordine = 0 OR (SELECT "fermata?"
  FROM sezione
```

```
  WHERE sezione.percorso = sez1.percorso
  AND sezione.ordine = sez1.ordine - 1))
--Comando per trovare il binario (visto che si riferisce alla sezione
precendente (che potrebbe anche essere in un altro
--spostamento del tutto)
AND (binario.ordine, binario.percorso) IN (SELECT sez2.ordine, sez2.percorso
              FROM sezione sez2, percorso p2, spostamento s2
              WHERE
                 (   sez2.percorso = sez1.percorso
                 AND sez1.ordine > 0
                 AND sez2.tratta_fine = stazione)
              OR
                 (   sez1.ordine = 0
                 AND p2.identificativo = (SELECT p3.identificativo
                          FROM percorso p3, spostamento s3
                          WHERE s3.treno = t1.identificativo
                          AND p3.identificativo = s3.percorso
                          AND p3.ora_arrivo < p1.ora_partenza
                          AND p3.ora_arrivo >= ALL (SELECT p4.ora_arrivo
                                       FROM percorso p4, spostamento s4
                                       WHERE p4.identificativo = s4.percorso
                                       AND s4.treno = t1.identificativo
                                       AND p4.ora_arrivo < p1.ora_partenza))
                                    AND sez2.ordine = (SELECT max(sez3.ordine)
                                    FROM sezione sez3
                                    WHERE sez3.percorso = sez2.percorso))))
AND (s1.giorno > current_date OR (s1.giorno = current_date AND
sez1.entrata_programmata > current_time))
ORDER BY s1.giorno, sez1.entrata_programmata ASC;
END

$$;
```

Calcolo Retribuzione di un Impiegato

```
create function calcolo_retribuzione_impiegato(cfnew codice_fiscale,
inizio_calcolo date) returns bigint
    language plpgsql
as
$$
DECLARE
        somma int8;
    BEGIN
    SELECT SUM(extract(HOUR from (t.fine-t.inizio))) * (SELECT c.paga_oraria
                                    FROM contratto c,
                                    WHERE cfnew=c.impiegato
                                    AND c.rescisso = false
                                    AND c.fine >= Inizio_Calcolo
                                    AND c.inizio <= Inizio_Calcolo) INTO somma
    from turno t, contratto c
    where extract(MONTH FROM t.giorno) = extract(MONTH FROM Inizio_Calcolo)
    AND t.impiegato = c.impiegato
    AND t.giorno <= Inizio_Calcolo
    AND t.giorno >= c.inizio
    AND t.giorno <= c.fine
    AND t.impiegato = cfnew;
    RETURN somma;
    end;
$$;
```