Oefeningen op Datamodel Examen

Hoofdstuk 4

Selecteer het ticket_nr, de client_id en de price uit CLIENT_PACK. Geef de prijs in het volgende formaat: $12.35. Zorg ervoor dat er geen prijzen weergegeven worden tussen $15 en $20, sorteer alles op prijs in dalende volgorde. Geef de prijskolom weer als 'Formatted'.

```
SELECT ticket_nr, client_id, TO_CHAR(price, '$99,99') AS "Formatted"
FROM client_pack
WHERE price NOT BETWEEN 15 AND 20
ORDER BY 3;
```

130 rows selected

Gebruik de SYSDATE en FROM DUAL om volgende boodschap op het scherm te toveren: Thursday's exam is easy! (Dit is de uitkomst moest je het op de dag van het examen runnen, doe je dit op woensdag dan is het 'Wednesday's exam is easy!')

```
SELECT TO_CHAR(SYSDATE, 'fmDay') || '''s exam is easy!'
FROM DUAL;
```

Gebruik de delivery_date uit CLIENT_PACK om volgend resultaat te bekomen wanneer de delivery date 20-09-2014 is: The packet was delivered at the Twentieth of September in 2014. Geef de kolom weer als 'Delivery date'.

```
SELECT 'The packet was delivered at the ' ||
TO_CHAR(delivery_date, 'fmDdspth "of" Month "in" YYYY') AS "Delivery date"
FROM client_pack;
```

Geef alle pakketen geleverd na 2013, geef ook de pickup_location.

```
SELECT delivery_date, pickup_location
FROM client_pack
WHERE delivery_date > TO_DATE('01-JAN-2014', 'DD-MON-YYYY');
```

Creëer een unieke tag door gebruik te maken van firstname, lastname, street en housenr uit CLIENT. De tag moet bestaan uit de 3 eerste letters van de voornaam, 3 laatste letters van de achternaam, 2 eerste letters van de straat en het huisnummer. Er mogen geen spaties voorkomen in de tag. Zorg ervoor dat alles in hoofdletters staat.

```
SELECT UPPER(CONCAT(CONCAT(CONCAT(SUBSTR(firstname, 0, 3), SUBSTR(lastname,-3, 3)),
SUBSTR(street, 0, 3)), housenr)) AS "Nestception"
FROM client;
```

TRUSTELIE12 en PIESELWAT48 moeten voorkomen in het resultaat.

Oefeningen op Datamodel Examen

Selecteer departure_loc en country_code uit COUNTRY_PACK. Gebruik een CASE om alle country_code om te vormen naar hun volledige naam (indien je een country code niet kent, gebruik je onze vriend Google). Geef deze kolom de naam 'Land'. Sorteer op departure_loc in aflopende volgorde.

```
SELECT departure_loc,
CASE country_code
WHEN 'BE' THEN 'Belgium'
WHEN 'PT' THEN 'Portugal'
WHEN 'IT' THEN 'Italy'
WHEN 'FR' THEN 'France'
WHEN 'GE' THEN 'Germany'
WHEN 'TR' THEN 'Kalkoen(Turkey)'
ELSE country_code END AS "Land"
FROM country_pack
ORDER BY departure_loc DESC;
```

Zelfde opdracht als hierboven, maar gebruik nu DECODE in plaats van CASE.

```
SELECT departure_loc,
DECODE (country_code,
'BE','Belgium',
'PT', 'Portugal',
'IT', 'Italy',
'FR', 'France',
'GE', 'Germany',
'TR', 'Kalkoen(Turkey)',
country_code ) AS "Land"
FROM country_pack
ORDER BY departure_loc DESC;
```

Oefeningen op Datamodel Examen

Hoofdstuk 5

Gebruik de price uit CLIENT_PACK. Geef het gemiddelde, maximum, minimum en som van de price. Zorg dat het gemiddelde er alsvolgt uitziet: $1,234.56 indien de prijs 123456 zou zijn. Deze kolom noem je 'Gemiddelde'. Rond het maximum af met als naam 'Afgerond'. Kap het minimum af met als naam 'Afgekapt'. Pas op de som een LPAD toe waarin je met * opvult zodat er altijd 10 tekens zijn opgevuld. Noem deze kolom 'LPADDED SUM'.

```
SELECT TO_CHAR(AVG(price), '$9,999.99') AS "Gemiddelde", ROUND(MAX(price)) AS
"Afgerond", TRUNC(MIN(price)) AS "Afgekapt", LPAD(SUM(price), 10, '*') AS "LPADDED SUM"
FROM client_pack;
```

Ga verder op vorige oplossing, zorg nu dat er een prompt komt met de vraag voor welke pickup_location dit uitgevoerd moet worden. Zorg ervoor dat de gebruiker geen foute invoer kan doen.

```
SELECT TO_CHAR(AVG(price), '$9,999.99') AS "Gemiddelde", ROUND(MAX(price)) AS
"Afgerond", TRUNC(MIN(price)) AS "Afgekapt", LPAD(SUM(price), 10, '*') AS "LPADDED SUM"
FROM client_pack
WHERE UPPER(pickup_location) IN (UPPER('&pickup_location'));
```

Gebruik TRANSPORT_PACK om erachter te komen wanneer het eerste pakket verzonden is en wanneer het laatste pakket aangekomen is.

```
SELECT MIN(departure_date), MAX(arrival_date)
FROM transport_pack;
```

19-JUL-08 en 08-DEC-11 respectievelijk

Kom te weten hoeveel keer 'BE' voorkomt in COUNTRY_PACK.

```
SELECT COUNT(country_code)
FROM country_pack
WHERE country_code LIKE 'BE';
```

Kom via CLIENT_PACK te weten wat het totaalbedrag is per klant dat ze gespendeerd hebben, op deze manier weet je aan welke klant je meer aandacht moet geven en aan welke klant minder aandacht. Sorteer op het totaalbedrag, geef deze kolom weer als 'Totaal' en zorg dat de klant die het meest heeft uitgegeven, bovenaan de lijst komt.

```
SELECT client_id, SUM(price) AS "Totaal"
FROM client_pack
GROUP BY client_id
ORDER BY 2 DESC;
```

GA VERDER OP VORIGE OPGAVE, ZORG DAT JE WEET HOEVEEL ELKE KLANT GESPENDEERD HEEFT BINNEN DE PRIJSKLASSE VAN 100 TOT EN MET 250.

```
SELECT client_id, SUM(price)
FROM client_pack
GROUP BY client_id
HAVING SUM(price) BETWEEN 100 AND 250
ORDER BY 2 DESC
```

Hoofdstuk 6

GEEF CLIENT_REGNR, FIRSTNAME, CITY EN POSTAL_CODE. GEBRUIK HIERVOOR EEN NATURAL JOIN.

```
SELECT client_regnr, firstname, city, postal_code
FROM client
NATURAL JOIN city;
```

ZELFDE OPGAVE ALS HIERBOVEN, GEBRUIK DEZE KEER EEN JOIN MET USING CLAUSE.

```
SELECT client_regnr, firstname, city, postal_code
FROM client JOIN city
USING (city);
```

GEEF CITY, POSTAL_CODE, COUNTRY_CODE EN COUNTRY_NAME. MAAK HIERVOOR GEBRUIK VAN EEN JOIN MET ON CLAUSE. JE MOET VOOR ELKE COUNTRY_NAME GEWOON DE COUNTRY_NAME GEVEN, BEHALVE VAN 'TURKEY', HIERVOOR GEEF JE 'KALKOEN' TERUG IN PLAATS VAN DE COUNTRY_NAME. DE COUNTRY_NAME KOLOM GEEF JE DE NAAM 'LAND'.

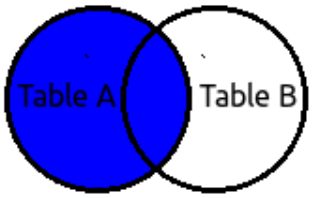
```
SELECT city.city, city.postal_code, city.country_code,
CASE country.country_name
WHEN 'Turkey' THEN 'Kalkoen'
ELSE country.country_name END AS "Land"
FROM city city JOIN country country
ON (city.country_code = country.country_code);
```

Oefeningen op Datamodel Examen

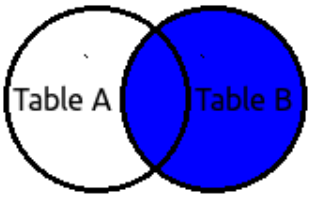
GEEF DEPARTURE_LOCATION, COUNTRY_CODE VAN DE DEPARTURE_LOCATION, ARRIVAL_LOCATION, COUNTRY_CODE VAN DE ARRIVAL_LOCATION EN COUNTRY_NAME VAN DE ARRIVAL_LOCATION. ZORG DAT JE BELGIË ALS ARRIVAL_LOCATION ERUIT FILTERT (DIT KAN DOORMIDDEL VAN COUNTRY_CODE OF COUNTRY_NAME VAN ARRIVAL_LOCATION). TIP: 4-WAY JOIN!

```
SELECT tp.departure_location, cp.country_code, tp.arrival_location, ci.country_code,
co.country_name
FROM transport_pack tp JOIN country_pack cp
ON (tp.departure_location = cp.departure_loc)
JOIN city ci
ON (tp.arrival_location = ci.city)
JOIN country co
ON (ci.country_code = co.country_code)
WHERE ci.country_code NOT LIKE 'BE';
```

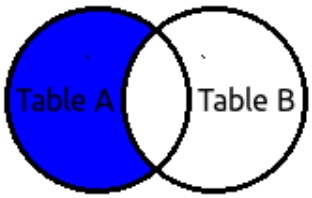
Documentatie voor JOINS.



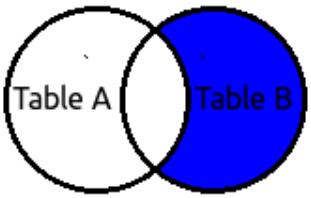

```
SELECT [list] FROM
    [Table A] A
LEFT JOIN
    [Table B] B
ON A.Value = B.Value
```



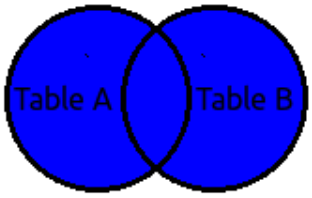

```
SELECT [list] FROM
    [Table A] A
RIGHT JOIN
    [Table B] B
ON A.Value = B.Value
```



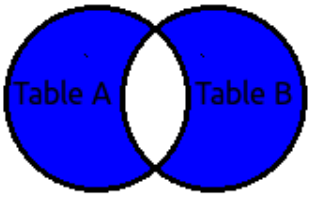

```
SELECT [list] FROM
    [Table A] A
LEFT JOIN
    [Table B] B
ON A.Value = B.Value
WHERE B.Value IS NULL
```



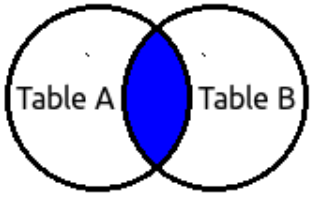

```
SELECT [list] FROM
    [Table A] A
RIGHT JOIN
    [Table B] B
ON A.Value = B.Value
WHERE A.Value IS NULL
```




```
SELECT [list] FROM
    [Table A] A
FULL OUTER JOIN
    [Table B] B
ON A.Value = B.Value
```




```
SELECT [list] FROM
    [Table A] A
FULL OUTER JOIN
    [Table B] B
ON A.Value = B.Value
WHERE A.Value IS NULL
OR B.Value IS NULL
```




```
SELECT [list] FROM
    [Table A] A
INNER JOIN
    [Table B] B
ON A.Value = B.Value
```