

RĪGAS TEHNISKĀ UNIVERSITĀTE

Datorzinātnes un informācijas tehnoloģijas fakultāte

1.praktiskais darbs

mācību priekšmetā

“Pēcrelāciju datu bāzes sistēmas(DB2)”

**Daudzdimensiju datu bāze.**

Izstrādāja: Viktors Stepičevs

III.kurss RDBD0, 4.grupa, 191RDB100

Pārbaudīja: Prof. J. Eiduks

**2021./22./māc.gads**

**Darba saturs**

[**Darba uzdevumi** 3](#_Toc84696759)

[**Anotācija** 4](#_Toc84696760)

[**Daudzdimensiju datu bāzes projektēšana** 5](#_Toc84696761)

[**Dimensiju definēšana** 7](#_Toc84696762)

[**Dinamiska datu analīzes uzdevuma realizācija** 10](#_Toc84696763)

[**1.** **vaicājums** 10](#_Toc84696764)

[**2.** **vaicājums** 10](#_Toc84696765)

[**3.** **vaicājums** 10](#_Toc84696766)

[**4.** **vaicājums** 11](#_Toc84696767)

[**5.** **vaicājums** 12](#_Toc84696768)

[**6.** **vaicājums** 13](#_Toc84696769)

[**7.** **vaicājums** 13](#_Toc84696770)

[**8.** **vaicājums** 14](#_Toc84696771)

[**Materializēta skata definēšana** 16](#_Toc84696772)

[**Secinājumi** 18](#_Toc84696773)

[**Literatūra** 19](#_Toc84696774)

[**Pielikumi** 20](#_Toc84696775)

[**Python skripts faktu tabulas ievades datiem** 20](#_Toc84696776)

[**SQL\*Loader command file** 21](#_Toc84696777)

[**Darbā izmantotas datu bāzes SQL kods** 21](#_Toc84696778)

# **Darba uzdevumi**

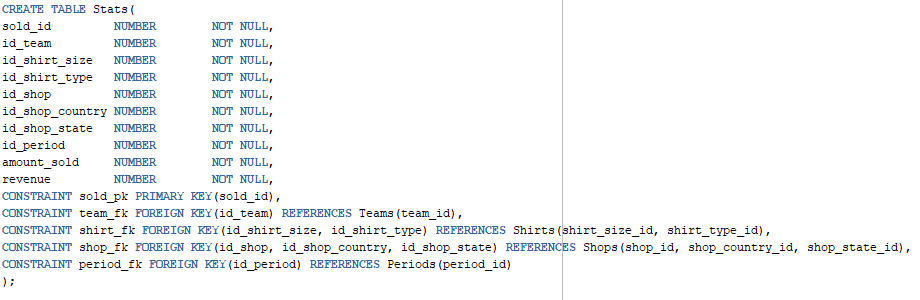
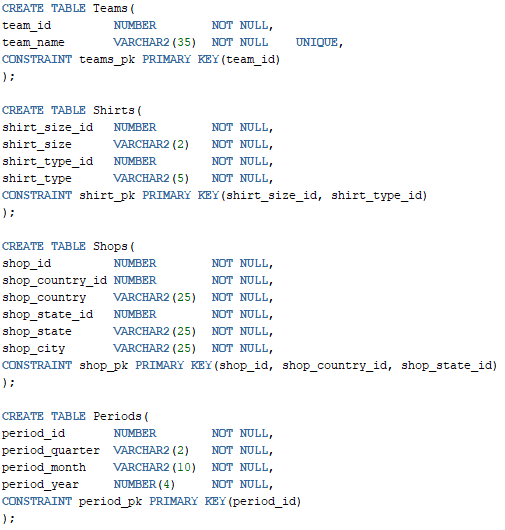
1. Jāveic daudzdimensiju datu bāzes projektēšana (faktu tabula, dimensiju tabulas (ar hierarhijām)) un jāievada dati tabulās (programma SQL\*Loader).  
2. Jādefinē dimensijas (create dimension …) un jāpārbauda to izveide.  
3. Jāizdomā un jārealizē dināmisks datu analīzes uzdevums ( vismaz 7 vaicājumi).  
4. Jādefinē materializētais skats un jāpārliecinās par tā darbību (vai notiek vaicājuma pārrakstīšanas (rewriting) process, izmantojot Explain plan programmu.  
5. Secinājumi par darbu (nevis kas ir izdarīts, bet kas noskaidrots un kādi vērtējumi apskatītai tehnoloģijai).

# **Anotācija**

Darba uzdevumu izpildes laikā tiks projektētā daudzdimensiju datu bāzes sistēma sporta veikalam, definējot vairākas dimensiju tabulas un faktu tabulu. Darba gaitā tiks definētas dimensijas un materializētais skats, ka arī tiks realizēts dinamisks datu analīzes uzdevums. Datu bāze tika izstrādātā Oracle SQL Developer, vienas tabulu datu ģenerēšanai tikai izmantota Python valoda.

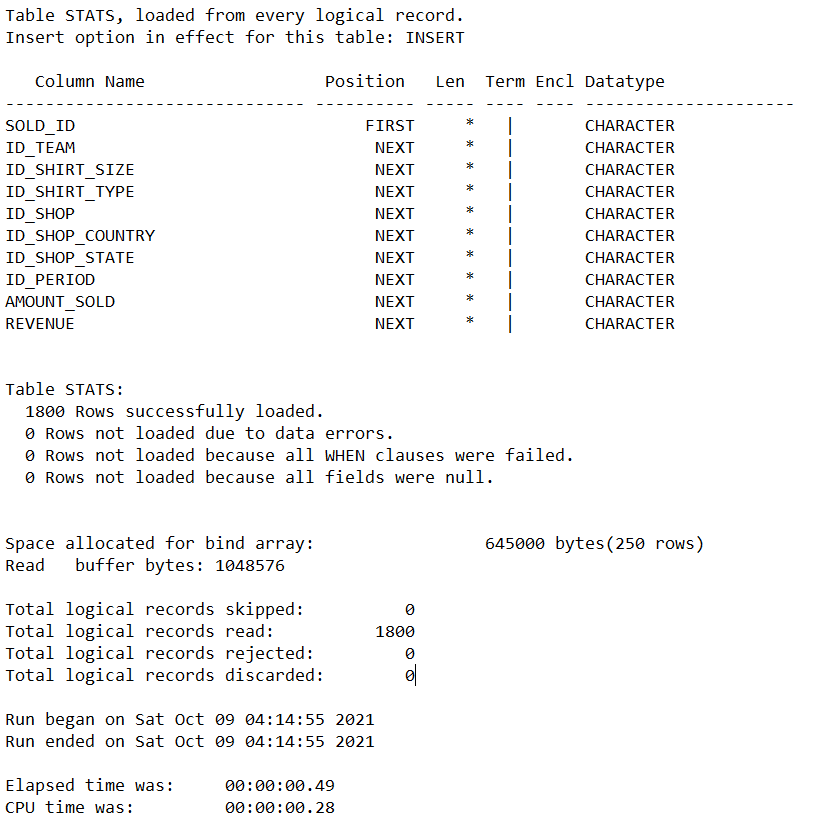
# **Daudzdimensiju datu bāzes projektēšana**

Uzsākot datu bāzes projektēšanu vispirms ir jādefinē problemsfēru noteiktajai tabulu struktūrai. Par pamatu šī darba izveidei es pieņēmu lēmumu turpināt iepriekšējā semestrā izmantoto sportisko tematiku, tajā procesā izveidojot 5 tabulas (Teams, Shirts, Shops, Periods, Stats). Būtībā sakot tiek definēta datu bāzē, kura apkopo statistiku par kāda sporta veikala pārdotiem futbola krekliem(kreklu pārdotais skaits un veikala peļņa).



Kopumā, četrās no piecām tabulām dati tika ievadīti ierastā veidā, izmantojot INSERT INTO komandu, bet tabulai faktu tabulai Stats būtu vērsts pievērst uzmanību. Lai turpmāk šajā daudzdimensiju datu bāzē būtu ko analizēt, bet ar roku bez kļūdām pārrakstīt visus indeksus būtu ļoti sarežģīti, es uzņemos uzģenerēt indeksus un statistiskas vērtības izmantojot Python valodu. Arī viens no iemesliem, kurā dēļ man rādījās tāda ideja bija tas, ka pēc uzdevuma bija jāievada datus izmantojot SQL\*Loader programmu. Izanalizējot to darbības principu, es sapratu, ka ir iespējams uzģenerēt vērtības .txt vai .csv tabulā un pa tiešo pārnest tās vērtības tabulā.

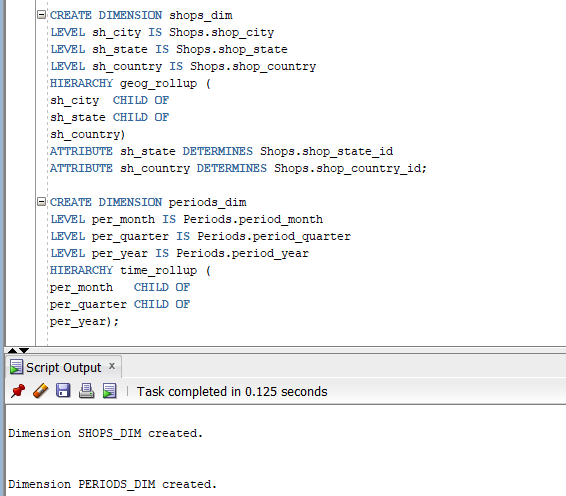
Darbā izvēlējos apstāties pie piecām komandām, kuru kreklu statistika tiks apkopota. Tālāk, kreklu klasifikācija tika sadalīta uz divām pozīcijām: kreklu izmērs un tips(mūsdienās, vairākām komandām tiek veidoti trīs dažādi spēles formu komplekti), kurām katrām tika piešķirs unikāls identifikators. Faktu tabulā šis ārējās atslēgās tiks ievietotas pēc gadījuma, lai rezultāta visiem krekliem būtu dažādais pārdotais daudzums, lai dati būtu reālistiskākie. Statistika tika apkopota 10 veikaliem dažādos valstīs, reģionos un pilsētās, trīs gadu robežā. Apkopojot to informāciju, uzģenerēju indeksus tādā veidā, ka katram no veikaliem būtu informācija par katru komandu katrā no analizētājiem laika posmiem. Izveidojot visas ārējās atslēgas, tālāk ar gadījuma skaitļiem tika piešķirtas vērtības veikala peļņu un kreklu pārdota skaitļa kolonnām. Pārdoto kreklu skaita kolonna tika aizpildīta ar vērtībām diapazonā no 1000 līdz 3000, bet peļņu kolonna ar vērtībām no 75000 līdz 100000. Tādās vērtības es paņēmu tādēļ, kā tādā gadījumā vismazākā iespējamā kreklu vidēja vērtība būtu 25, bet vislielākā 100, kas kopumā ir diezgan reālistiski, ņemot vērā, ka veikalos varētu būt kādā veidā speciālie piedāvājumi vai kaut kas tām līdzīgs.



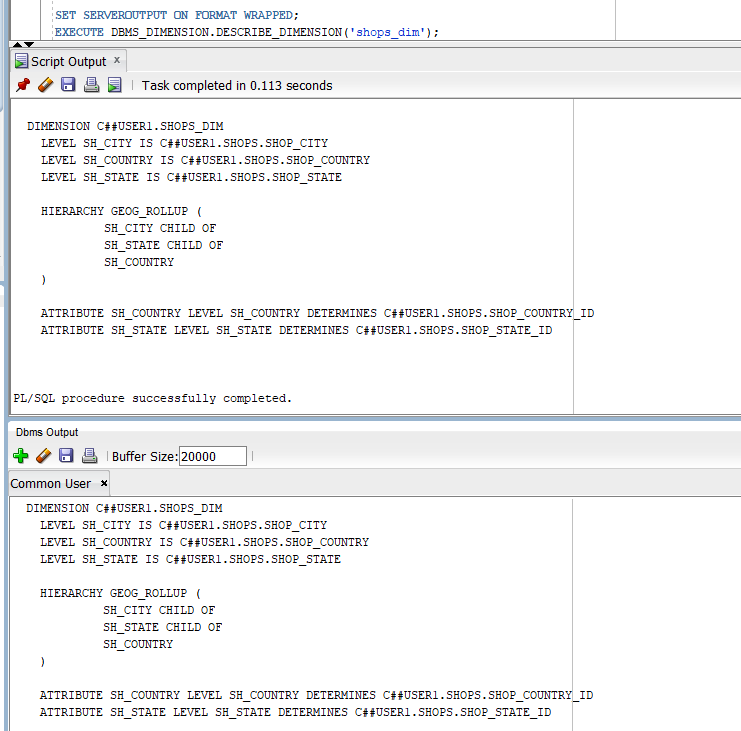
Visi darbā pielietotie skripti var būt atrodamie darba pielikumā.

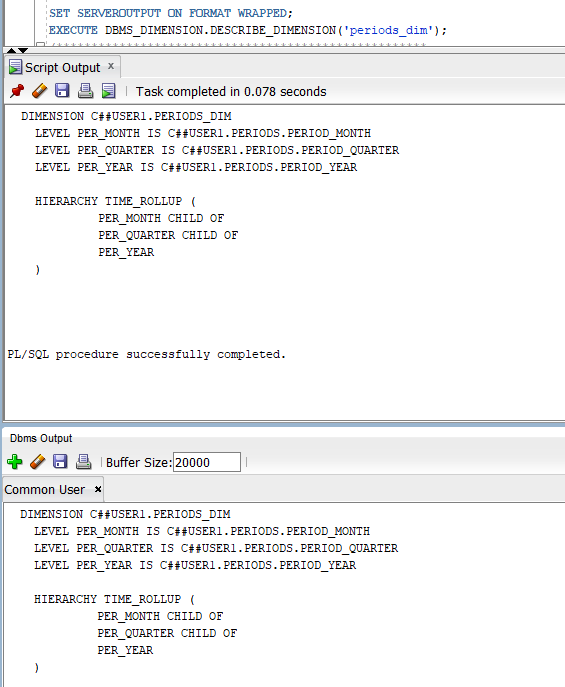
# **Dimensiju definēšana**

Definējot dimensijas manas datu bāzes kontekstā viennozīmīgi izcēlās divas dimensiju tabulas ar hierarhijām. Shops tabulā ir novērota ģeogrāfiska hierarhija ar pilsētām, valsts rajoniem un pati valsti. Starp citu šajā pašā tabulā rajoniem un valstīm ir īpašais identifikatora atribūts, kuru attiecības ir iespējams atspoguļot izmantojot ATTRIBUTE DETERMINES konstrukciju. Otrā tabula ir Periods tabula, kurā ir novēlojama laika mēru hierarhija. Kā ir redzams, dimensijas tika veiksmīgi izveidotas.



Pēc dimensiju izveides, to izveides procesu ir iespējams pārbaudīt ar DESCRIBE\_DIMENSION procedūras palīdzību. Kā ir redzams attēlā, rezultātu arī ir iespējams novērot DBMS Output buferī, jo procedūra DESCRIBE\_DIMENSION ir daļa no DBMS\_DIMENSION paketes, kā arī rezultātos arī ir iespējams novērot lietotāju, ar kuru tika veidots savienojums ar datu bāzi.





# **Dinamiska datu analīzes uzdevuma realizācija**

Ņemot vērā definēto tabulu struktūru ir vairākās iespējas dažādā veidā analizēt datus. (Vaicājumus likšu teksta veidā, lai būtu labāk redzams)

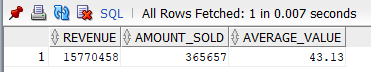
## **vaicājums**

**SELECT SUM(revenue) AS revenue, SUM(amount\_sold) AS amount\_sold, ROUND(SUM(revenue) / SUM(amount\_sold), 2) AS average\_value**

**FROM Stats, Shops, Periods**

**WHERE (shop\_country = 'England' AND period\_year = 2019) AND period\_id = id\_period AND shop\_id = id\_shop AND shop\_country\_id = id\_shop\_countryAND shop\_state\_id = id\_shop\_state;**

Ar pirmā vaicājuma palīdzību tika atrasta kopēja peļņa un pārdotājs kreklu skaits veikaliem no Anglijas 2019. gada laikā, pielietojot agregācijas funkcijas SUM. Ar agregācijas funkcijas AVG palīdzīgu tika atrasta vidēja krekla cena, noapaļojot rezultātu līdz 2 zīmēm aiz komata, ņemot vērā to faktu, ka tā ir kreklu cena.



## **vaicājums**

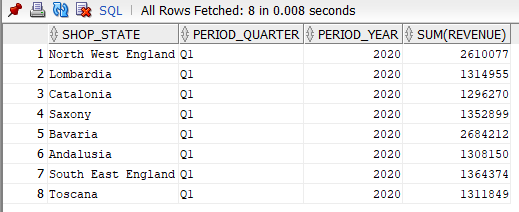
**SELECT shop\_state, period\_quarter, period\_year, SUM(revenue)**

**FROM Shops, Periods, Stats**

**WHERE period\_quarter = 'Q1' AND period\_year = 2020 AND shop\_id = id\_shop AND shop\_state\_id = id\_shop\_state AND shop\_country\_id = id\_shop\_country AND period\_id = id\_period**

**GROUP BY shop\_state, period\_quarter, period\_year;**

Otrajā vaicājuma tika atrasta kopēja peļņa 2020. gada pirmajām kvartālam pa valsts reģioniem ar GROUP BY klauzulas palīdzību



## **vaicājums**

**SELECT shop\_country, shirt\_size, shirt\_type, SUM(amount\_sold) AS amount**

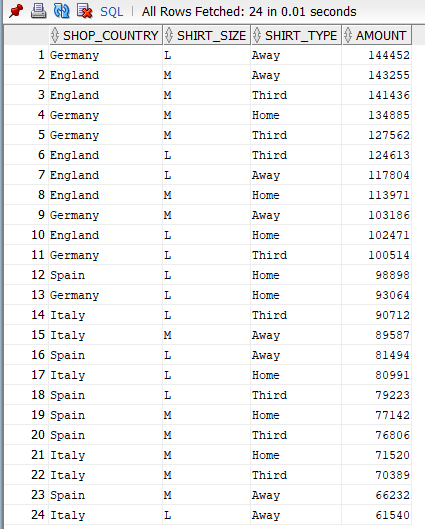
**FROM Shops, Shirts, Stats**

**WHERE shirt\_size IN ('M', 'L') AND shop\_id = id\_shop AND shop\_state\_id = id\_shop\_state AND shop\_country\_id = id\_shop\_country AND shirt\_size\_id = id\_shirt\_size AND shirt\_type\_id = id\_shirt\_type**

**GROUP BY shop\_country, shirt\_size, shirt\_type**

**ORDER BY amount DESC;**

Trešā vaicājuma jēga ir sakārtot kreklu pārdošanas statistiku par L un M izmēra krekliem, neatkarīgi no tā tipa. Ar ORDER BY klauzulas palīdzību dati tiek sakārtoti ar DESC(dilstošā secībā) paplašinājuma palīdzību.



## **vaicājums**

**SELECT team\_name, shirt\_type, period\_year, SUM(amount\_sold) AS amount**

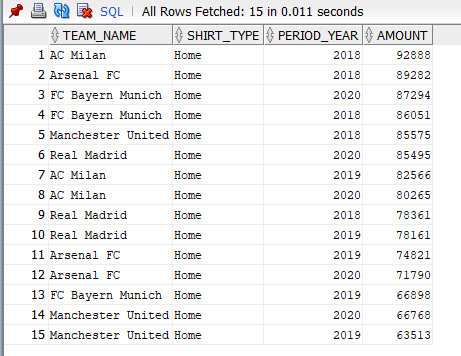
**FROM Teams, Shirts, Periods, Stats**

**WHERE shirt\_type = 'Home' AND team\_id = id\_team AND shirt\_size\_id = id\_shirt\_size AND shirt\_type\_id = id\_shirt\_type AND period\_id = id\_period**

**GROUP BY team\_name, shirt\_type, period\_year**

**ORDER BY amount DESC;**

Nākamais vaicājums ļauj mums saprast, kuras “mājas” krekli bija vispieprasītākie katrā gadā.



## **vaicājums**

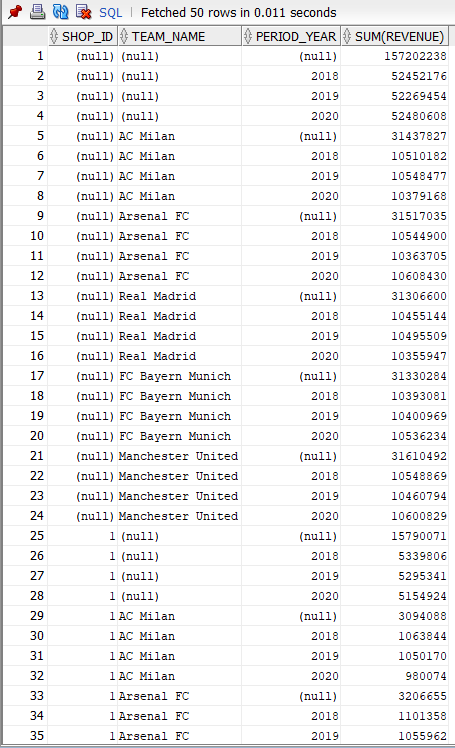
**SELECT shop\_id, team\_name, period\_year, SUM(revenue)**

**FROM Shops, Teams, Periods, Stats**

**WHERE shop\_id = id\_shop AND shop\_country\_id = id\_shop\_country AND shop\_state\_id = id\_shop\_state AND team\_id = id\_team AND period\_id = id\_period**

**GROUP BY CUBE(shop\_id, team\_name, period\_year);**

Sekojoša vaicājumā tika izvadīta kopēja statistika par peļņu atkarībā no vairākiem parametriem. Tika pielietota GROUP BY klauzulas CUBE paplašinājums, kur vispirms tika izvadīta informācija par kopējo peļņu novērojamā laikā, tālāk katrā no aplūkojamiem gadiem, un tālāk katrai komandai kārta gadā. Kā ir iespējams novērot, katrai komandai arī tiek ģenerēta datu starpsumma, kura rezultē peļņu no katras komandas krekliem. Ar nākama vaicājuma palīdzību būs vieglāk saprast GROUP BY CUBE() konstrukcijas darbību. (bildē nav visi izvades dati)



**…**

## **vaicājums**

**SELECT shop\_id, team\_name, period\_year, SUM(revenue),**

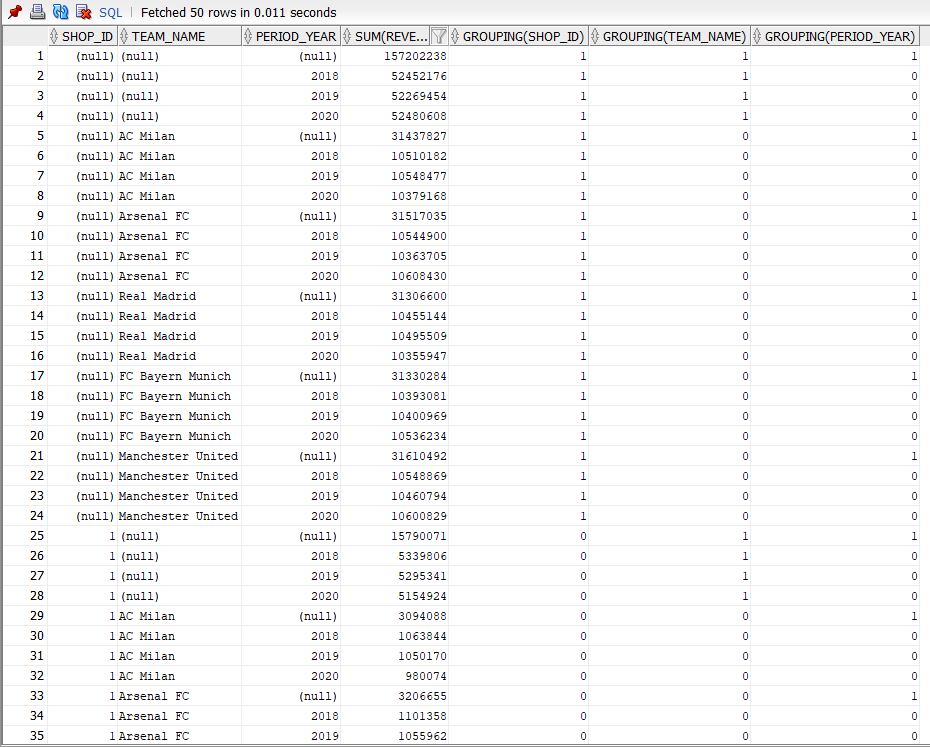
**GROUPING(shop\_id), GROUPING(team\_name), GROUPING(period\_year)**

**FROM Shops, Teams, Periods, Stats**

**WHERE shop\_id = id\_shop AND shop\_country\_id = id\_shop\_country AND shop\_state\_id = id\_shop\_state AND team\_id = id\_team AND period\_id = id\_period**

**GROUP BY CUBE(shop\_id, team\_name, period\_year);**

Būtībā, pievienojot GROUPING funkciju tagad ir skaidri saprotams, kur un pēc kāda parametra tiek ģenerētas starpvērtības.

****

**…**

## **vaicājums**

**SELECT shop\_id, team\_name, shirt\_size, SUM(amount\_sold) AS sold,**

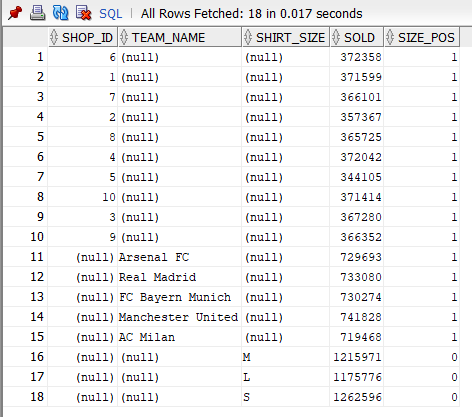
**GROUPING(shirt\_size) AS size\_pos**

**FROM Shops, Teams, Shirts, Stats**

**WHERE shop\_id = id\_shop AND shop\_country\_id = shop\_country\_id AND shop\_state\_id = id\_shop\_state AND team\_id = id\_team AND shirt\_size\_id = id\_shirt\_size AND shirt\_type\_id = id\_shirt\_type**

**GROUP BY GROUPING SETS(shop\_id, team\_name, shirt\_size);**

Grupējot datus ar GROUPING SETS() palīdzību, salīdzinājuma ar GROUP BY CUBE konstrukciju, šeit tiek izvadīta kopējā pieprasīta informācija par noteikto parametra vērtību. Šajā vaicājumā tiek izvadītā informācija par pārdoto kreklu skaitu katram veikalā, komandai un izmērām.



## **vaicājums**

**WITH**

**Avg\_shirt\_revenue AS**

**(SELECT team\_name, ROUND(AVG(revenue), 2) AS average\_shirt\_revenue**

**FROM Teams, Stats**

**WHERE team\_id = id\_team**

**GROUP BY team\_name),**

**Avg\_sold AS**

**(SELECT team\_name, ROUND(AVG(amount\_sold), 0) AS average\_shirts\_sold**

**FROM Teams, Stats**

**WHERE team\_id = id\_team**

**GROUP BY team\_name)**

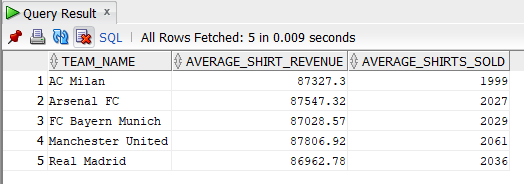
**SELECT P.team\_name, P.average\_shirt\_revenue, A.average\_shirts\_sold**

**FROM Avg\_shirt\_revenue P, Avg\_sold A**

**WHERE P.team\_name = A.team\_name**

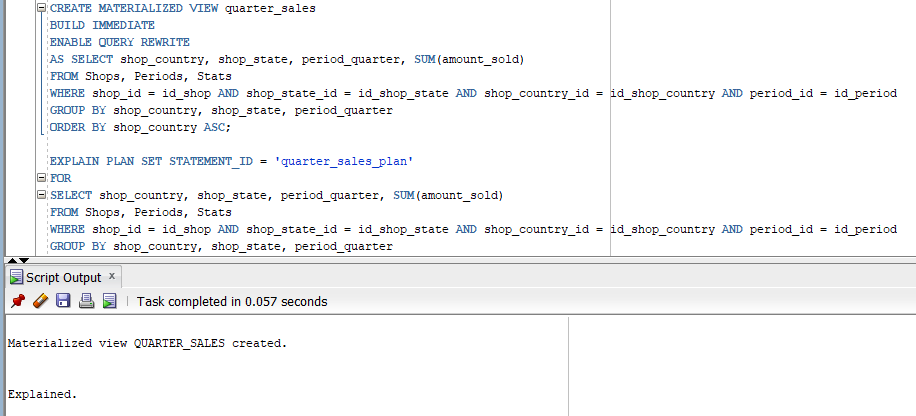
**ORDER BY team\_name ASC;**

Ar šī vaicājuma palīdzību izmantojot WITH klauzulu tika rekursīvi iegūta vidēja vērtība no ikgadējas kreklu peļņas un vidējais pārdoto kreklu skaits.

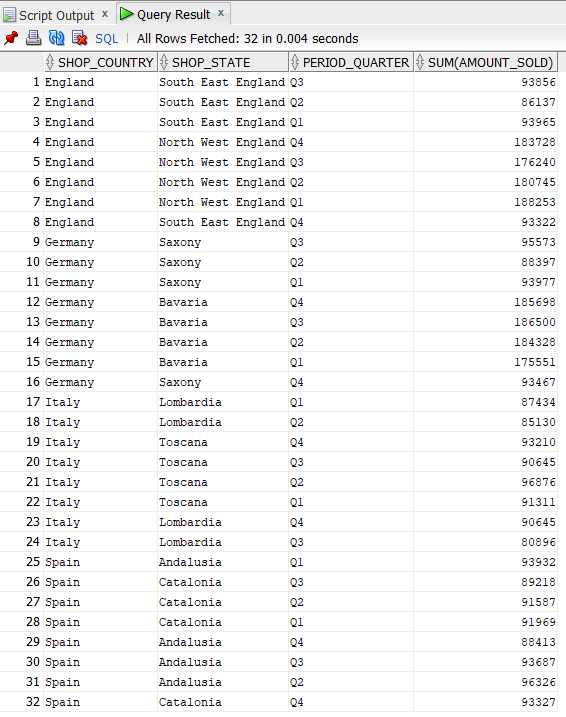


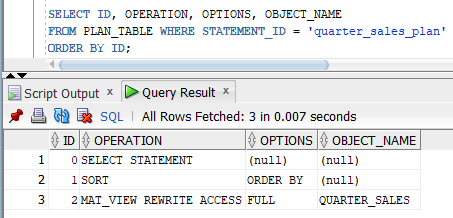
# **Materializēta skata definēšana**

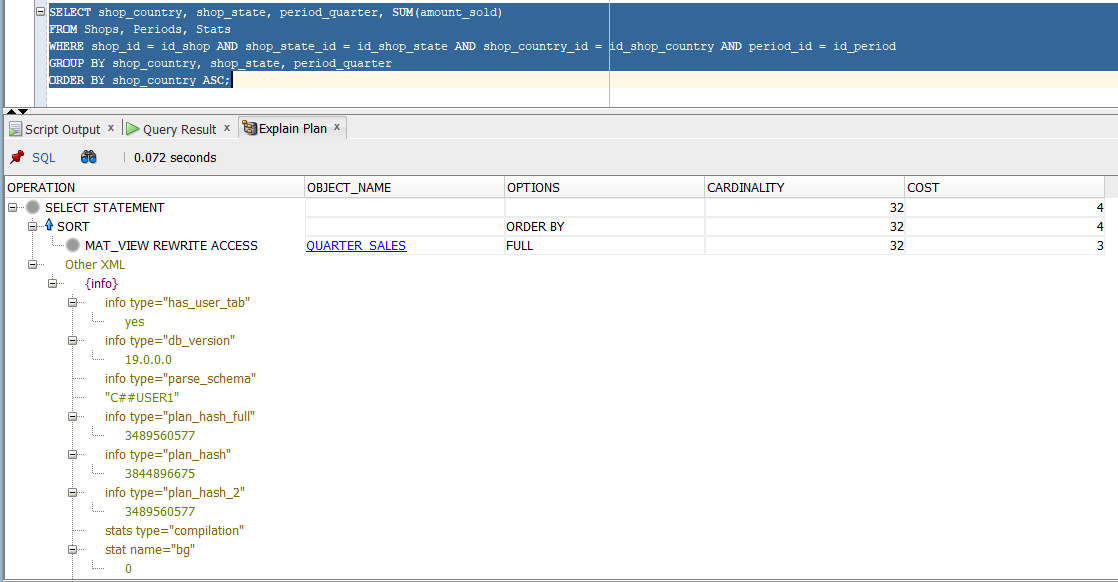
Būtība sākot, materializētais skats ir kādā definēta vaicājuma atbildes kopija, kas ļauj samazināt izpildes laiku vaicājumiem ar plašu atribūtu agregāciju.



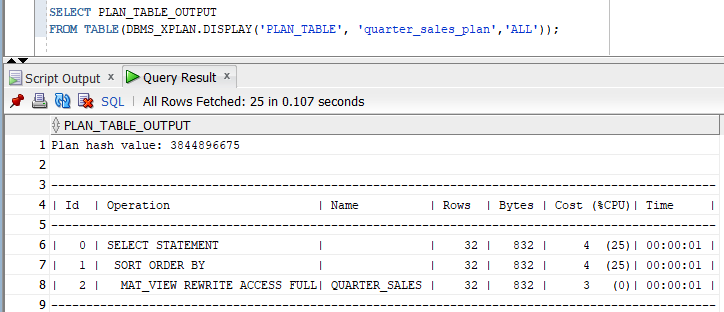
Piemērām izveidoju materializēto skatu vaicājumam, kurš izvada kopējo pārdoto kreklu skaitu katrām reģionam atsevišķajos gada kvartālos.







Ieguvot datu agregācijas rezultātus ir redzams, ka SELECT vaicājums izmanto CPU resursus, ko iespējams novērot izdarot tā, lai materializētais skats veidotu datu agregāciju iepriekš, tātad vaicājums izmantos iepriekš agregētus datus no materializēta skata, nevis tērējot esošos datora resursus.



# **Secinājumi**

Analizējot panākto darba uzdevumu izpildes gaitā rezultātu, manuprāt, darbs klājās diezgan veiksmīgi. Šī darba izpildei man parādījās iespēja praktiski apskatīt tādās konstrukcijas kā materializētājs skats un dimensijas. Kopumā, bija vieglāk pildīt šo darbu ņemot vērā, ka pagājušajā semestra laikā es jau strādāju ar Oracle SQL Developer rīku, kas palīdzēja vieglāk un ātrāk uzsākt darba izpildes gaitu.

Godīgi sakot, es varu teikt, ka darba izpildes laika man radījās dažas problēmas, kuras bija kārtīgi jāatrisina. Tagad es varu teikt, kā problēmās, kuras bija sastopamas šī darba izpildes gaitā visticamāk nākotnē mani nekaitēs, jo būtībā vairāk problēmu bija nevis ar uzdevumu izprašanu vai realizēšanu, bet vairāk ar Oracle konfigurāciju, pat neskatoties uz to faktu, ka iepriekš viss strādāja kārtīgi. Piemērām, bija problēmās ielādēt datu faili tabulā caur SQL\*Loader programmu. Pirmo laiku sistēma ļoti ilgi apstrādāja ar mani izveidoto failu ar vērtībām, bet patrenējoties ar citiem datiem un tabulām arī man palīdzēja labāk izprast šo darbību, un pēc tām sistēmai apstrādāt manu teksta failu aizņēma 1 sekundi laika. Arī bija dažas problēmas ar to teksta faila ģenerēšanu, jo vajadzēja kārtīgi izplānot kādām vērtībām un kad ir jāmijiedarbojas, lai iegūtu loģisko rezultātu.

Kopumā, personīgi es biju domājis, kā šīs darbs man panāksies ātrāk, nekā tas tika pabeigts, bet pēc manām domām patērēto laiku varētu uzskatīt par lietderīgo, jo darba izpildes laikā, lai panāktu nepieciešamo rezultātu man vajadzēja risināt problēmas ar nestandartiem paņēmieniem, kurus es varēšu pielietot nākošajos darbos.

# **Literatūra**

1. Prof. J. Eiduks. Lekciju konspekti priekšmetā “Pēcrelāciju datu bāzes sistēmas (DB2)”
2. <https://datubaze.wordpress.com/>
3. <https://docs.oracle.com/cd/B19306_01/server.102/b14215/ldr_concepts.htm>
4. <https://www.oracletutorial.com/oracle-administration/oracle-sqlloader/>
5. <https://docs.oracle.com/cd/B19306_01/server.102/b14200/statements_5006.htm>
6. <https://docs.oracle.com/en/database/oracle/oracle-database/21/sqlrf/CREATE-DIMENSION.html#GUID-E6CD4CFC-5D06-4A8F-9DF1-C609A7EB8413>
7. <https://docs.oracle.com/cd/B19306_01/server.102/b14200/statements_6002.htm>
8. <https://docs.oracle.com/database/121/DWHSG/dimen.htm#DWHSG8265>
9. <https://docs.oracle.com/cd/B19306_01/server.102/b14200/statements_6002.htm>

# **Pielikumi**

## **Python skripts faktu tabulas ievades datiem**

import random as rand

file = open("load.txt", 'w')

id\_team = 0

id\_shop = 0

id\_shop\_country = 0

id\_shop\_state = 0

id\_period = 0

for i in range(1, 1801):

if i % 360 == 1:

id\_team += 1

id\_shop = 0

if i % 36 == 1:

id\_shop += 1

id\_period = 0

if id\_shop == 1 or id\_shop == 2 or id\_shop == 3:

id\_shop\_country = 1

elif id\_shop == 4 or id\_shop == 5:

id\_shop\_country = 2

elif id\_shop == 6 or id\_shop == 7:

id\_shop\_country = 3

elif id\_shop == 8 or id\_shop == 9 or id\_shop == 10:

id\_shop\_country = 4

if id\_shop == 1 or id\_shop == 2:

id\_shop\_state = 1

elif id\_shop == 3:

id\_shop\_state = 2

elif id\_shop == 4:

id\_shop\_state = 3

elif id\_shop == 5:

id\_shop\_state = 4

elif id\_shop == 6:

id\_shop\_state = 5

elif id\_shop == 7:

id\_shop\_state = 6

elif id\_shop == 8 or id\_shop == 9:

id\_shop\_state = 7

elif id\_shop == 10:

id\_shop\_state = 8

id\_period += 1

amount\_sold = rand.randint(1000, 3000)

revenue = rand.randint(75000, 100000)

id\_shirt\_size = rand.randint(1, 3)

id\_shirt\_type = rand.randint(1, 3)

file.write(str(i) + "|" + str(id\_team) + "|" + str(id\_shirt\_size) + "|" + str(id\_shirt\_type) + "|" + str(id\_shop) + "|" + str(id\_shop\_country) + "|" + str(id\_shop\_state) + "|" + str(id\_period) + "|" + str(amount\_sold) + "|" + str(revenue) + "\n")

file.close()

## **SQL\*Loader command file**

LOAD DATA

INFILE 'C:\Users\testdb\Desktop\1. pr\data.txt'

INTO TABLE Stats

fields terminated by "|"

(

sold\_id,

id\_team,

id\_shirt\_size,

id\_shirt\_type,

id\_shop,

id\_shop\_country,

id\_shop\_state,

id\_period,

amount\_sold,

revenue

)

## **Darbā izmantotas datu bāzes SQL kods**

DROP TABLE Stats;

DROP TABLE Teams;

DROP TABLE Shirts;

DROP TABLE Shops;

DROP TABLE Periods;

CREATE TABLE Teams(

team\_id NUMBER NOT NULL,

team\_name VARCHAR2(35) NOT NULL UNIQUE,

CONSTRAINT teams\_pk PRIMARY KEY(team\_id)

);

CREATE TABLE Shirts(

shirt\_size\_id NUMBER NOT NULL,

shirt\_size VARCHAR2(2) NOT NULL,

shirt\_type\_id NUMBER NOT NULL,

shirt\_type VARCHAR2(5) NOT NULL,

CONSTRAINT shirt\_pk PRIMARY KEY(shirt\_size\_id, shirt\_type\_id)

);

CREATE TABLE Shops(

shop\_id NUMBER NOT NULL,

shop\_country\_id NUMBER NOT NULL,

shop\_country VARCHAR2(25) NOT NULL,

shop\_state\_id NUMBER NOT NULL,

shop\_state VARCHAR2(25) NOT NULL,

shop\_city VARCHAR2(25) NOT NULL,

CONSTRAINT shop\_pk PRIMARY KEY(shop\_id, shop\_country\_id, shop\_state\_id)

);

CREATE TABLE Periods(

period\_id NUMBER NOT NULL,

period\_quarter VARCHAR2(2) NOT NULL,

period\_month VARCHAR2(10) NOT NULL,

period\_year NUMBER(4) NOT NULL,

CONSTRAINT period\_pk PRIMARY KEY(period\_id)

);

CREATE TABLE Stats(

sold\_id NUMBER NOT NULL,

id\_team NUMBER NOT NULL,

id\_shirt\_size NUMBER NOT NULL,

id\_shirt\_type NUMBER NOT NULL,

id\_shop NUMBER NOT NULL,

id\_shop\_country NUMBER NOT NULL,

id\_shop\_state NUMBER NOT NULL,

id\_period NUMBER NOT NULL,

amount\_sold NUMBER NOT NULL,

revenue NUMBER NOT NULL,

CONSTRAINT sold\_pk PRIMARY KEY(sold\_id),

CONSTRAINT team\_fk FOREIGN KEY(id\_team) REFERENCES Teams(team\_id),

CONSTRAINT shirt\_fk FOREIGN KEY(id\_shirt\_size, id\_shirt\_type) REFERENCES Shirts(shirt\_size\_id, shirt\_type\_id),

CONSTRAINT shop\_fk FOREIGN KEY(id\_shop, id\_shop\_country, id\_shop\_state) REFERENCES Shops(shop\_id, shop\_country\_id, shop\_state\_id),

CONSTRAINT period\_fk FOREIGN KEY(id\_period) REFERENCES Periods(period\_id)

);

INSERT INTO Teams VALUES('1', 'Manchester United');

INSERT INTO Teams VALUES('2', 'AC Milan');

INSERT INTO Teams VALUES('3', 'Real Madrid');

INSERT INTO Teams VALUES('4', 'Arsenal FC');

INSERT INTO Teams VALUES('5', 'FC Bayern Munich');

INSERT INTO Shirts VALUES('1', 'S', '1', 'Home');

INSERT INTO Shirts VALUES('1', 'S', '2', 'Away');

INSERT INTO Shirts VALUES('1', 'S', '3', 'Third');

INSERT INTO Shirts VALUES('2', 'M', '1', 'Home');

INSERT INTO Shirts VALUES('2', 'M', '2', 'Away');

INSERT INTO Shirts VALUES('2', 'M', '3', 'Third');

INSERT INTO Shirts VALUES('3', 'L', '1', 'Home');

INSERT INTO Shirts VALUES('3', 'L', '2', 'Away');

INSERT INTO Shirts VALUES('3', 'L', '3', 'Third');

INSERT INTO Shops VALUES('1', '1', 'England', '1', 'North West England', 'Manchester');

INSERT INTO Shops VALUES('2', '1', 'England', '1', 'North West England', 'Liverpool');

INSERT INTO Shops VALUES('3', '1', 'England', '2', 'South East England', 'Southampton');

INSERT INTO Shops VALUES('4', '2', 'Italy', '3', 'Toscana', 'Florence');

INSERT INTO Shops VALUES('5', '2', 'Italy', '4', 'Lombardia', 'Milan');

INSERT INTO Shops VALUES('6', '3', 'Spain', '5', 'Andalusia', 'Seville');

INSERT INTO Shops VALUES('7', '3', 'Spain', '6', 'Catalonia', 'Barcelona');

INSERT INTO Shops VALUES('8', '4', 'Germany', '7', 'Bavaria', 'Munich');

INSERT INTO Shops VALUES('9', '4', 'Germany', '7', 'Bavaria', 'Nuremberg');

INSERT INTO Shops VALUES('10', '4', 'Germany', '8', 'Saxony', 'Dresden');

INSERT INTO Periods VALUES('1', 'Q1', 'January', '2018');

INSERT INTO Periods VALUES('2', 'Q1', 'February', '2018');

INSERT INTO Periods VALUES('3', 'Q1', 'March', '2018');

INSERT INTO Periods VALUES('4', 'Q2', 'April', '2018');

INSERT INTO Periods VALUES('5', 'Q2', 'May', '2018');

INSERT INTO Periods VALUES('6', 'Q2', 'June', '2018');

INSERT INTO Periods VALUES('7', 'Q3', 'July', '2018');

INSERT INTO Periods VALUES('8', 'Q3', 'August', '2018');

INSERT INTO Periods VALUES('9', 'Q3', 'September', '2018');

INSERT INTO Periods VALUES('10', 'Q4', 'October', '2018');

INSERT INTO Periods VALUES('11', 'Q4', 'November', '2018');

INSERT INTO Periods VALUES('12', 'Q4', 'December', '2018');

INSERT INTO Periods VALUES('13', 'Q1', 'January', '2019');

INSERT INTO Periods VALUES('14', 'Q1', 'February', '2019');

INSERT INTO Periods VALUES('15', 'Q1', 'March', '2019');

INSERT INTO Periods VALUES('16', 'Q2', 'April', '2019');

INSERT INTO Periods VALUES('17', 'Q2', 'May', '2019');

INSERT INTO Periods VALUES('18', 'Q2', 'June', '2019');

INSERT INTO Periods VALUES('19', 'Q3', 'July', '2019');

INSERT INTO Periods VALUES('20', 'Q3', 'August', '2019');

INSERT INTO Periods VALUES('21', 'Q3', 'September', '2019');

INSERT INTO Periods VALUES('22', 'Q4', 'October', '2019');

INSERT INTO Periods VALUES('23', 'Q4', 'November', '2019');

INSERT INTO Periods VALUES('24', 'Q4', 'December', '2019');

INSERT INTO Periods VALUES('25', 'Q1', 'January', '2020');

INSERT INTO Periods VALUES('26', 'Q1', 'February', '2020');

INSERT INTO Periods VALUES('27', 'Q1', 'March', '2020');

INSERT INTO Periods VALUES('28', 'Q2', 'April', '2020');

INSERT INTO Periods VALUES('29', 'Q2', 'May', '2020');

INSERT INTO Periods VALUES('30', 'Q2', 'June', '2020');

INSERT INTO Periods VALUES('31', 'Q3', 'July', '2020');

INSERT INTO Periods VALUES('32', 'Q3', 'August', '2020');

INSERT INTO Periods VALUES('33', 'Q3', 'September', '2020');

INSERT INTO Periods VALUES('34', 'Q4', 'October', '2020');

INSERT INTO Periods VALUES('35', 'Q4', 'November', '2020');

INSERT INTO Periods VALUES('36', 'Q4', 'December', '2020');

SELECT \* FROM Stats

ORDER BY sold\_id ASC;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Dimensijas

\*/

DROP DIMENSION shops\_dim;

DROP DIMENSION periods\_dim;

CREATE DIMENSION shops\_dim

LEVEL sh\_city IS Shops.shop\_city

LEVEL sh\_state IS Shops.shop\_state

LEVEL sh\_country IS Shops.shop\_country

HIERARCHY geog\_rollup (

sh\_city CHILD OF

sh\_state CHILD OF

sh\_country)

ATTRIBUTE sh\_state DETERMINES Shops.shop\_state\_id

ATTRIBUTE sh\_country DETERMINES Shops.shop\_country\_id;

SET SERVEROUTPUT ON FORMAT WRAPPED;

EXECUTE DBMS\_DIMENSION.DESCRIBE\_DIMENSION('shops\_dim');

CREATE DIMENSION periods\_dim

LEVEL per\_month IS Periods.period\_month

LEVEL per\_quarter IS Periods.period\_quarter

LEVEL per\_year IS Periods.period\_year

HIERARCHY time\_rollup (

per\_month CHILD OF

per\_quarter CHILD OF

per\_year);

SET SERVEROUTPUT ON FORMAT WRAPPED;

EXECUTE DBMS\_DIMENSION.DESCRIBE\_DIMENSION('periods\_dim');

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Vaicajumi

\*/

SELECT SUM(revenue) AS revenue, SUM(amount\_sold) AS amount\_sold, ROUND(SUM(revenue) / SUM(amount\_sold), 2) AS average\_value

FROM Stats, Shops, Periods

WHERE (shop\_country = 'England' AND period\_year = 2019) AND period\_id = id\_period AND shop\_id = id\_shop AND shop\_country\_id = id\_shop\_country AND shop\_state\_id = id\_shop\_state;

SELECT shop\_state, period\_quarter, period\_year, SUM(revenue)

FROM Shops, Periods, Stats

WHERE period\_quarter = 'Q1' AND period\_year = 2020 AND shop\_id = id\_shop AND shop\_state\_id = id\_shop\_state AND shop\_country\_id = id\_shop\_country AND period\_id = id\_period

GROUP BY shop\_state, period\_quarter, period\_year;

SELECT shop\_country, shirt\_size, shirt\_type, SUM(amount\_sold) AS amount

FROM Shops, Shirts, Stats

WHERE shirt\_size IN ('M', 'L') AND shop\_id = id\_shop AND shop\_state\_id = id\_shop\_state AND shop\_country\_id = id\_shop\_country AND shirt\_size\_id = id\_shirt\_size AND shirt\_type\_id = id\_shirt\_type

GROUP BY shop\_country, shirt\_size, shirt\_type

ORDER BY amount DESC;

SELECT team\_name, shirt\_type, period\_year, SUM(amount\_sold) AS amount

FROM Teams, Shirts, Periods, Stats

WHERE shirt\_type = 'Home' AND team\_id = id\_team AND shirt\_size\_id = id\_shirt\_size AND shirt\_type\_id = id\_shirt\_type AND period\_id = id\_period

GROUP BY team\_name, shirt\_type, period\_year

ORDER BY amount DESC;

SELECT shop\_id, team\_name, period\_year, SUM(revenue)

FROM Shops, Teams, Periods, Stats

WHERE shop\_id = id\_shop AND shop\_country\_id = id\_shop\_country AND shop\_state\_id = id\_shop\_state AND team\_id = id\_team AND period\_id = id\_period

GROUP BY CUBE(shop\_id, team\_name, period\_year);

SELECT shop\_id, team\_name, period\_year, SUM(revenue),

GROUPING(shop\_id), GROUPING(team\_name), GROUPING(period\_year)

FROM Shops, Teams, Periods, Stats

WHERE shop\_id = id\_shop AND shop\_country\_id = id\_shop\_country AND shop\_state\_id = id\_shop\_state AND team\_id = id\_team AND period\_id = id\_period

GROUP BY CUBE(shop\_id, team\_name, period\_year);

SELECT shop\_id, team\_name, shirt\_size, SUM(amount\_sold) AS sold,

GROUPING(shirt\_size) AS size\_pos

FROM Shops, Teams, Shirts, Stats

WHERE shop\_id = id\_shop AND shop\_country\_id = shop\_country\_id AND shop\_state\_id = id\_shop\_state AND team\_id = id\_team AND shirt\_size\_id = id\_shirt\_size AND shirt\_type\_id = id\_shirt\_type

GROUP BY GROUPING SETS(shop\_id, team\_name, shirt\_size);

WITH

Avg\_shirt\_revenue AS

(SELECT team\_name, ROUND(AVG(revenue), 2) AS average\_shirt\_revenue

FROM Teams, Stats

WHERE team\_id = id\_team

GROUP BY team\_name),

Avg\_sold AS

(SELECT team\_name, ROUND(AVG(amount\_sold), 0) AS average\_shirts\_sold

FROM Teams, Stats

WHERE team\_id = id\_team

GROUP BY team\_name)

SELECT P.team\_name, P.average\_shirt\_revenue, A.average\_shirts\_sold

FROM Avg\_shirt\_revenue P, Avg\_sold A

WHERE P.team\_name = A.team\_name

ORDER BY team\_name ASC;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Materializetais skats

\*/

DROP MATERIALIZED VIEW quarter\_sales;

CREATE MATERIALIZED VIEW quarter\_sales

BUILD IMMEDIATE

REFRESH FORCE

ON DEMAND

ENABLE QUERY REWRITE

AS SELECT shop\_country, shop\_state, period\_quarter, SUM(amount\_sold)

FROM Shops, Periods, Stats

WHERE shop\_id = id\_shop AND shop\_state\_id = id\_shop\_state AND shop\_country\_id = id\_shop\_country AND period\_id = id\_period

GROUP BY shop\_country, shop\_state, period\_quarter

ORDER BY shop\_country ASC;

EXEC DBMS\_STATS.gather\_table\_stats(USER, 'quarter\_sales');

EXPLAIN PLAN SET STATEMENT\_ID = 'quarter\_sales\_plan'

FOR

SELECT shop\_country, shop\_state, period\_quarter, SUM(amount\_sold)

FROM Shops, Periods, Stats

WHERE shop\_id = id\_shop AND shop\_state\_id = id\_shop\_state AND shop\_country\_id = id\_shop\_country AND period\_id = id\_period

GROUP BY shop\_country, shop\_state, period\_quarter

ORDER BY shop\_country ASC;

SELECT ID, OPERATION, OPTIONS, OBJECT\_NAME

FROM PLAN\_TABLE WHERE STATEMENT\_ID = 'quarter\_sales\_plan'

ORDER BY ID;

SELECT PLAN\_TABLE\_OUTPUT

FROM TABLE(DBMS\_XPLAN.DISPLAY('PLAN\_TABLE', 'quarter\_sales\_plan','ALL'));