

מיני פרויקט בבסיסי נתונים

שם הארגון: חנות

שם האגף: נשים

מגישות:

תהילה בן עזרא 323845321

tasaraf@g.jct.ac.il

מיכל יששכר 213686496

Michalir17@gmail.com

מיני פרויקט בבסיסי נתונים

תוכן עניינים

4	פרק א'
4	תיאור כללי:
4	הישויות במערכת והקשרים ביניהם:
5	ERD Diagram
5	DSD Diagram
7	יצירת הטבלאות ע"י createTable
9	פקודת DESC:
10	DROP_TABLE:
11	קובץ SELECT
12	הכנסת נתונים על ידי שלושת הדרכים המבוקשות:
12	הכנסת נתונים על ידי DATA GENERATOR:
13	הכנסת נתונים על ידי קוד בפיתון ופקודות INSERT:
14	הכנסת נתונים מקובץ:
16	יצירת קובץ INSERT
17	גיבוי
19	מחיקת הפרויקט
20	שחזור הפרויקט
21	פרק ב
21	Select query 1:
22	Select query 2:
23	Select query 3:
24	Select query 4:
25	Delete query 1:
26	Delete query 2:
27	Update query 1
28	Update query 2:
30	Queries with parameters:
31	1.query with data parameter
32	2.query with date parameter
33	3.query with hint parameter
33	4.query with name parameter
33	5.query with list parameter

מיני פרויקט בבסיסי נתונים

35 Constraints
35 1.alter table –CHECK
35 2.alter table-DEFAULT
37 שלב 3
37 #1 פרוצדורה
41 #2 פרוצדורה
44 #1 פונקציה
48 #2 פונקציה
50 #1 תכנית
52 #2 תכנית
55 שלב 4
55 :DSD DIAGRAM
56 :ERD DIAGRAM
57 DSD DIAGRAM בתצוגה נוספת
58 טבלת ERD משותף
59 טבלת DSD משותף
59 VIEWS

מיני פרויקט בבסיסי נתונים

פרק א'

תיאור כללי:

בפרויקט זה נממש מערכת ניהול מידע עבור חנות. המערכת מכילה ישויות שונות וקשרים ביניהם על מנת למפות את כל המידע בצורה מיטבית. מטרת הפרויקט היא לעזור לנהל את החנות בצורה יעילה ומתועדת היטב.

הישויות במערכת והקשרים ביניהם:

לקוחות – אנשים שקונים בחנות

עובדים – אנשים העובדים בחנות

ספקים – ספקי סחורה לחנות

מוצרים – המוצרים הנמכרים בחנות

קטגוריות – שמות הקטגוריות שעל פיהן נמייין את המוצרים שבחנות. לכל מוצר יש קטגוריה אחת לפחות.

הזמנות – הזמנת מוצרים חדשים בין העובדים לספקים

רכישות – רכישת מוצרים על ידי הלקוחות מהעובדים

Entities:

Client = Client_ID, Client_Name, Is_Club_Member

Worker = Worker_Id, Worker_Name, Start_of_Work_Date

Supplier = Supplier_Id, Supplier_Name, Region

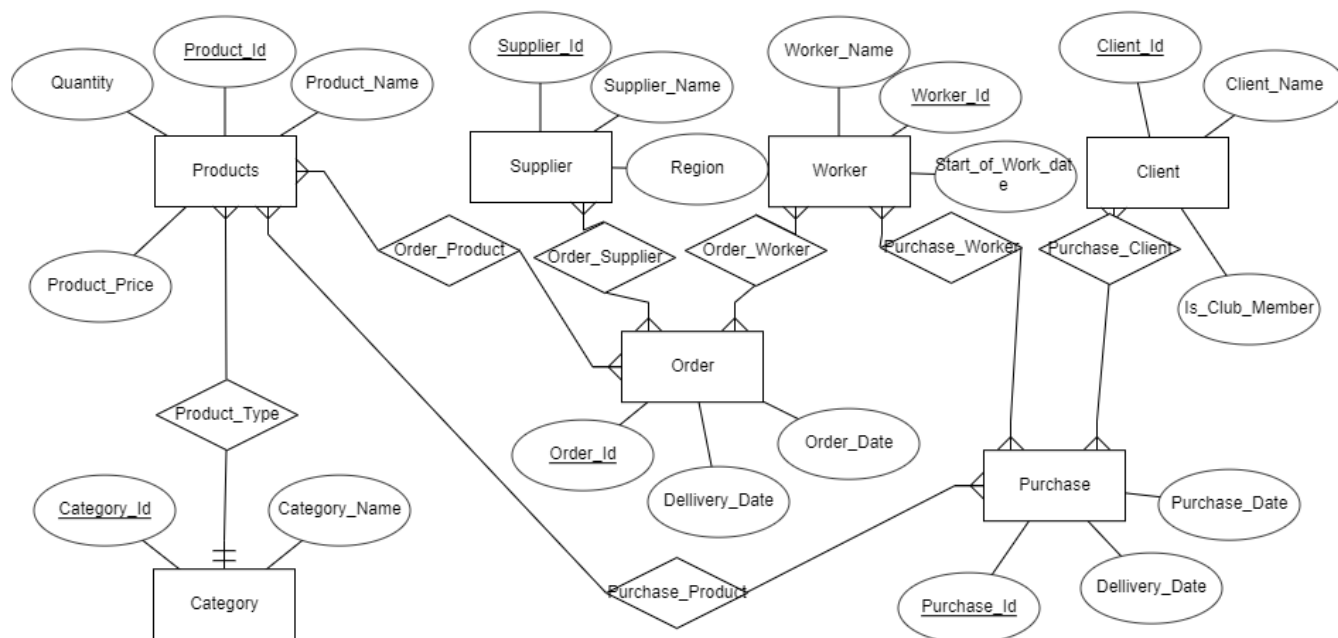
Products = Product_Id, Product_Name, Quantity

Category = Category_Id, Category_Name

Order = Order_Id, Order_Date, Delivery_Date, Quantity

Purchase = Purchase_Id, Purchase_Date, Delivery_Date, Quantity

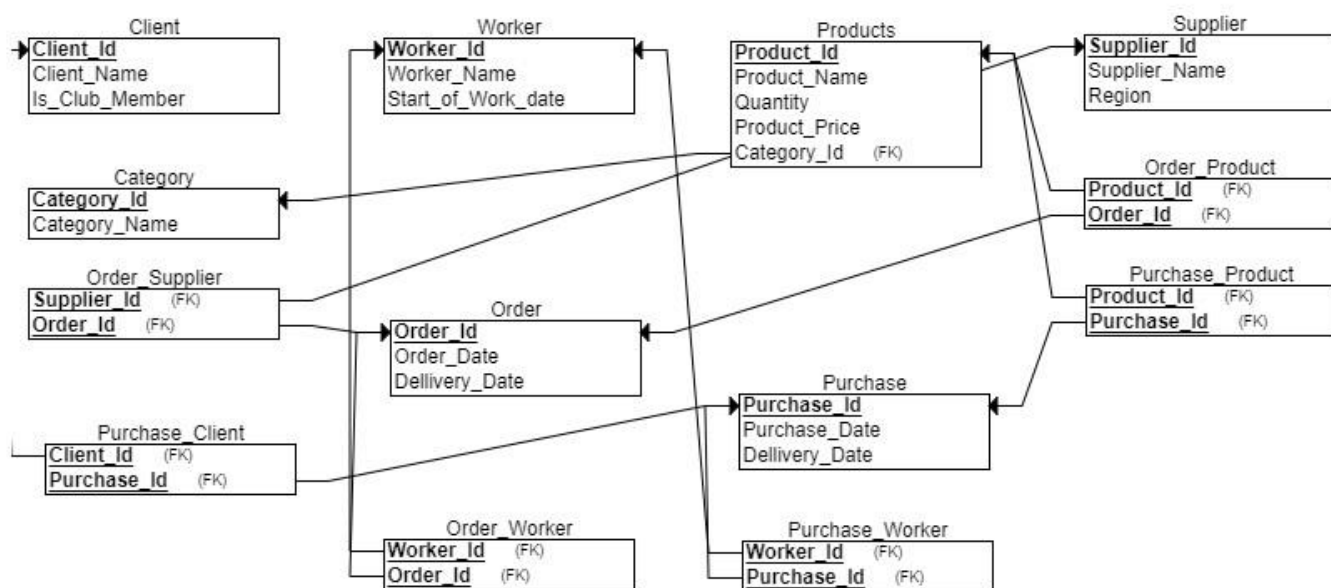
מיני פרויקט בבסיסי נתונים



ERD Diagram

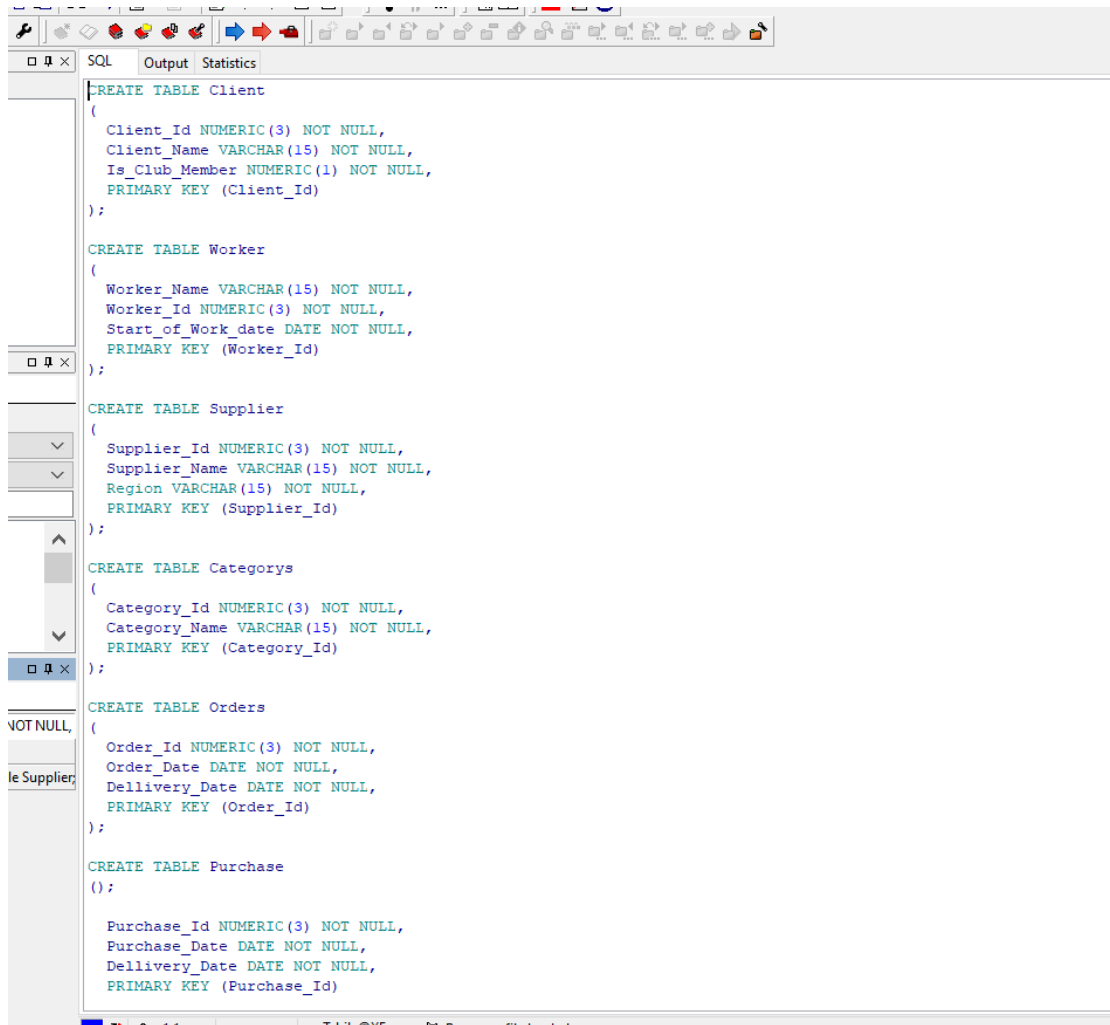
DSD Diagram

מיני פרויקט בבסיסי נתונים



מיני פרויקט בבסיסי נתונים

יצירת הטבלאות ע"י createTable



```
CREATE TABLE Client
(
    Client_Id NUMERIC(3) NOT NULL,
    Client_Name VARCHAR(15) NOT NULL,
    Is_Club_Member NUMERIC(1) NOT NULL,
    PRIMARY KEY (Client_Id)
);

CREATE TABLE Worker
(
    Worker_Name VARCHAR(15) NOT NULL,
    Worker_Id NUMERIC(3) NOT NULL,
    Start_of_Work_date DATE NOT NULL,
    PRIMARY KEY (Worker_Id)
);

CREATE TABLE Supplier
(
    Supplier_Id NUMERIC(3) NOT NULL,
    Supplier_Name VARCHAR(15) NOT NULL,
    Region VARCHAR(15) NOT NULL,
    PRIMARY KEY (Supplier_Id)
);

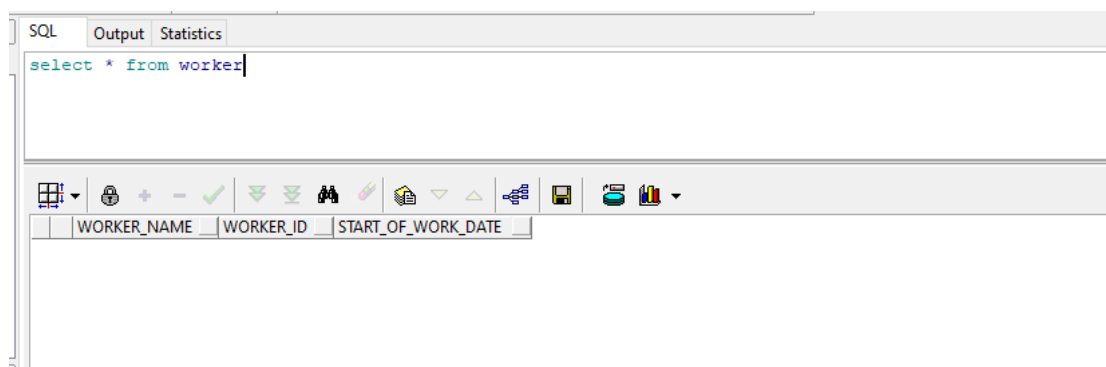
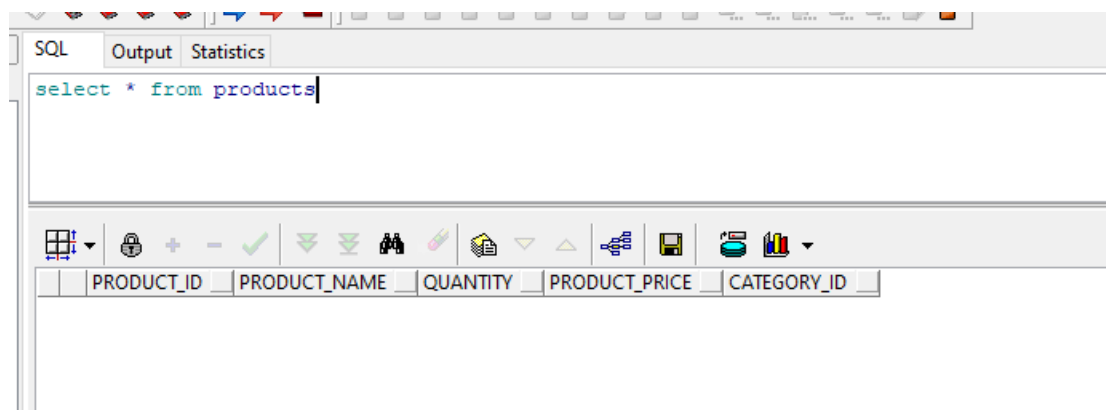
CREATE TABLE Categorys
(
    Category_Id NUMERIC(3) NOT NULL,
    Category_Name VARCHAR(15) NOT NULL,
    PRIMARY KEY (Category_Id)
);

CREATE TABLE Orders
(
    Order_Id NUMERIC(3) NOT NULL,
    Order_Date DATE NOT NULL,
    Dellivery_Date DATE NOT NULL,
    PRIMARY KEY (Order_Id)
);

CREATE TABLE Purchase
(
    Purchase_Id NUMERIC(3) NOT NULL,
    Purchase_Date DATE NOT NULL,
    Dellivery_Date DATE NOT NULL,
    PRIMARY KEY (Purchase_Id)
);
```

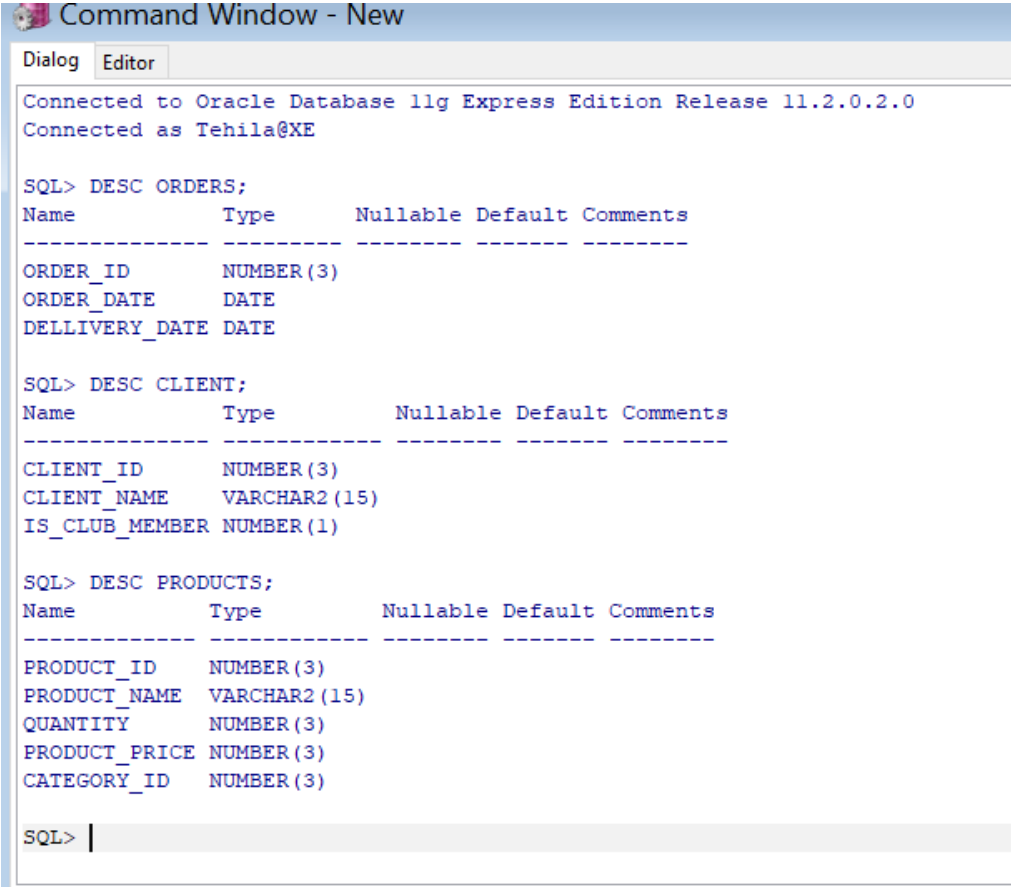
בדקנו שאכן הטבלאות נוצרו על ידי הרצה הסקריפט הזה:

מיני פרויקט בבסיסי נתונים



מיני פרויקט בבסיסי נתונים

פקודת DESC:



```
Command Window - New
Dialog Editor
Connected to Oracle Database 11g Express Edition Release 11.2.0.2.0
Connected as Tehila@XE

SQL> DESC ORDERS;
Name                Type                Nullable Default Comments
-----
ORDER_ID            NUMBER(3)
ORDER_DATE          DATE
DELLIVERY_DATE      DATE

SQL> DESC CLIENT;
Name                Type                Nullable Default Comments
-----
CLIENT_ID          NUMBER(3)
CLIENT_NAME        VARCHAR2(15)
IS_CLUB_MEMBER      NUMBER(1)

SQL> DESC PRODUCTS;
Name                Type                Nullable Default Comments
-----
PRODUCT_ID         NUMBER(3)
PRODUCT_NAME        VARCHAR2(15)
QUANTITY            NUMBER(3)
PRODUCT_PRICE       NUMBER(3)
CATEGORY_ID         NUMBER(3)

SQL> |
```

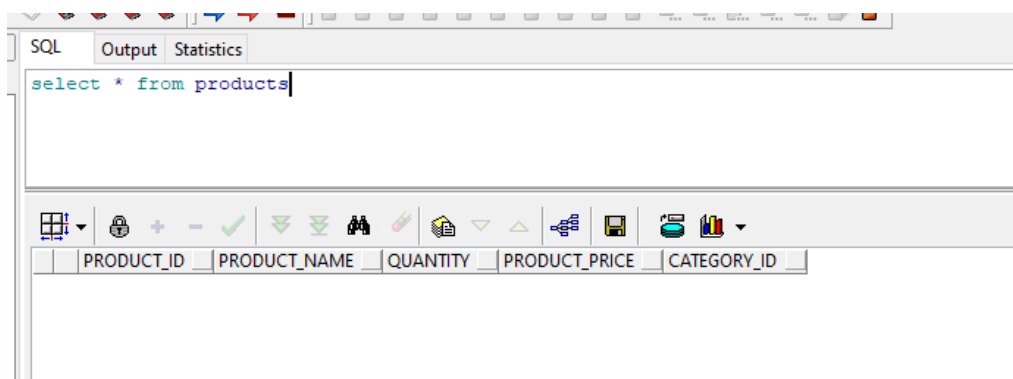
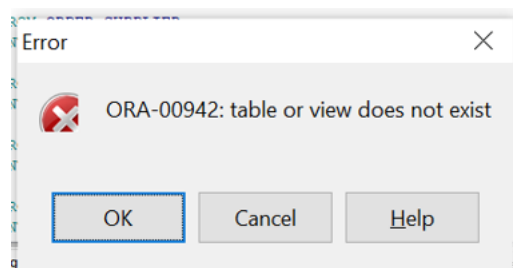
מיני פרויקט בבסיסי נתונים

:DROP_TABLE

כמובן על פי הסדר כך שהטבלאות התלויות בטבלאות אחרות נמחקות קודם

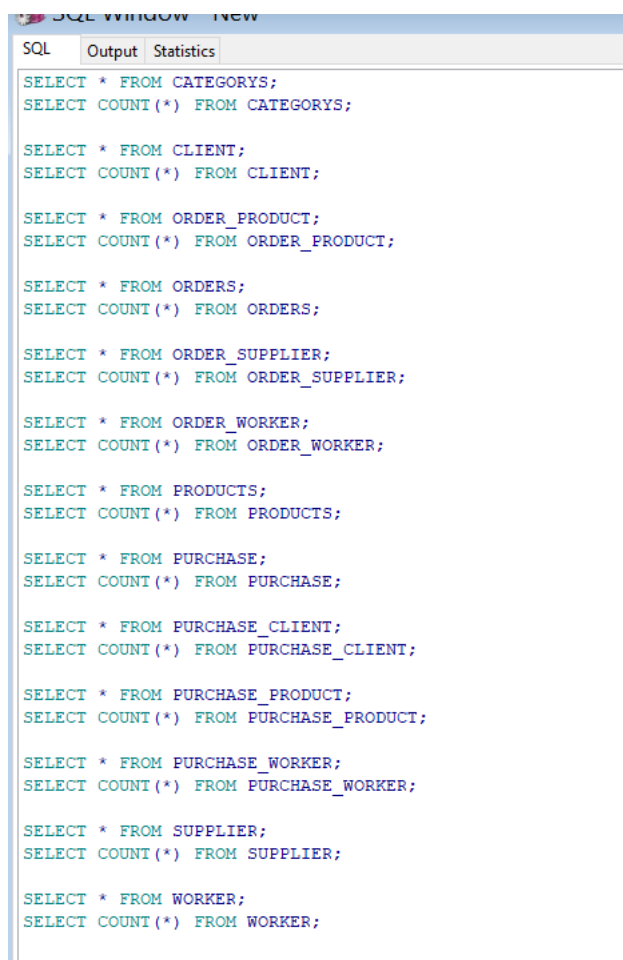
```
SQL Output Statistics
drop table Order_Product;
drop table Purchase_Product;
drop table Purchase_Client;
drop table Order_Supplier;
drop table Order_Worker;
drop table Purchase_Worker;
drop table Client;
drop table Worker;
drop table Supplier;
drop table Orders;
drop table Purchase;
drop table Products;
drop table Categorys;
```

בדקנו שהטבלאות נמחקות ולאחר מכן יצרנו אותן שוב מחדש



מיני פרויקט בבסיסי נתונים

קובץ SELECT



```
SQL Window New
SQL Output Statistics
SELECT * FROM CATEGORIES;
SELECT COUNT(*) FROM CATEGORIES;

SELECT * FROM CLIENT;
SELECT COUNT(*) FROM CLIENT;

SELECT * FROM ORDER_PRODUCT;
SELECT COUNT(*) FROM ORDER_PRODUCT;

SELECT * FROM ORDERS;
SELECT COUNT(*) FROM ORDERS;

SELECT * FROM ORDER_SUPPLIER;
SELECT COUNT(*) FROM ORDER_SUPPLIER;

SELECT * FROM ORDER_WORKER;
SELECT COUNT(*) FROM ORDER_WORKER;

SELECT * FROM PRODUCTS;
SELECT COUNT(*) FROM PRODUCTS;

SELECT * FROM PURCHASE;
SELECT COUNT(*) FROM PURCHASE;

SELECT * FROM PURCHASE_CLIENT;
SELECT COUNT(*) FROM PURCHASE_CLIENT;

SELECT * FROM PURCHASE_PRODUCT;
SELECT COUNT(*) FROM PURCHASE_PRODUCT;

SELECT * FROM PURCHASE_WORKER;
SELECT COUNT(*) FROM PURCHASE_WORKER;

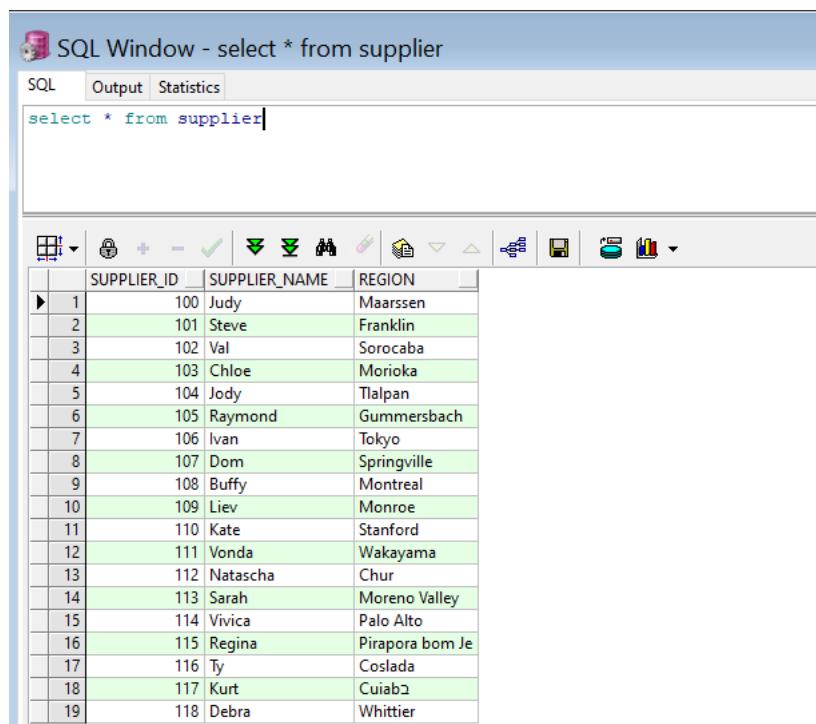
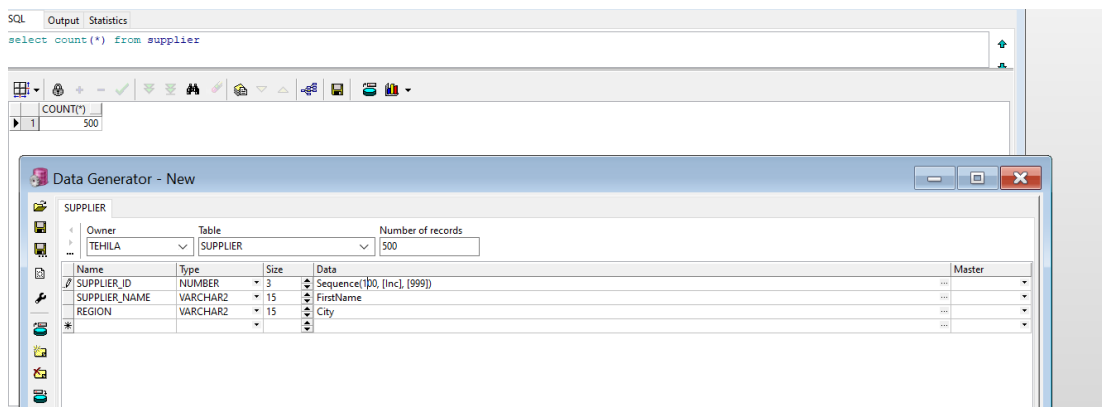
SELECT * FROM SUPPLIER;
SELECT COUNT(*) FROM SUPPLIER;

SELECT * FROM WORKER;
SELECT COUNT(*) FROM WORKER;
```

מיני פרויקט בבסיסי נתונים

הכנסת נתונים על ידי שלושת הדרכים המבוקשות:

הכנסת נתונים על ידי DATA GENERATOR:



מיני פרויקט בבסיסי נתונים

הכנסת נתונים על ידי קוד בפיתוח ופקודות INSERT:

```
def generate_insert_statement(existing_ids):
    purchase_id = generate_unique_purchase_id(existing_ids)
    existing_ids.add(purchase_id)
    purchase_date = generate_random_date()
    dellivey_date = generate_random_date()
    return f"INSERT INTO TEHILA.purchase(PURCHASE_ID, PURCHASE_DATE, DELLIVERY_DATE) VALUES({purchase_id}, (TO_DATE('{purchase_id}'

if __name__ == "__main__":
    # Set to keep track of existing IDs
    existing_ids = set()

    # Generate multiple insert statements
    for _ in range(400): # Change the range to generate more or fewer statements
        print(generate_insert_statement(existing_ids))
```

```
"C:\Users\Tehila Benezra\PycharmProjects\test2022b\venv\Scripts\python.exe" C:\assembly\xordll.py
INSERT INTO TEHILA.purchase(PURCHASE_ID, PURCHASE_DATE, DELLIVERY_DATE) VALUES(560, (TO_DATE('2027/10/03', 'yyyy/mm/dd')), (TO_DATE('2030/07/04', 'yyyy/mm/dd')));
INSERT INTO TEHILA.purchase(PURCHASE_ID, PURCHASE_DATE, DELLIVERY_DATE) VALUES(949, (TO_DATE('2025/07/29', 'yyyy/mm/dd')), (TO_DATE('2024/06/18', 'yyyy/mm/dd')));
INSERT INTO TEHILA.purchase(PURCHASE_ID, PURCHASE_DATE, DELLIVERY_DATE) VALUES(207, (TO_DATE('2024/01/29', 'yyyy/mm/dd')), (TO_DATE('2028/10/27', 'yyyy/mm/dd')));
INSERT INTO TEHILA.purchase(PURCHASE_ID, PURCHASE_DATE, DELLIVERY_DATE) VALUES(248, (TO_DATE('2025/06/08', 'yyyy/mm/dd')), (TO_DATE('2030/08/27', 'yyyy/mm/dd')));
INSERT INTO TEHILA.purchase(PURCHASE_ID, PURCHASE_DATE, DELLIVERY_DATE) VALUES(668, (TO_DATE('2027/12/27', 'yyyy/mm/dd')), (TO_DATE('2029/03/19', 'yyyy/mm/dd')));
INSERT INTO TEHILA.purchase(PURCHASE_ID, PURCHASE_DATE, DELLIVERY_DATE) VALUES(740, (TO_DATE('2030/08/19', 'yyyy/mm/dd')), (TO_DATE('2025/07/25', 'yyyy/mm/dd')));
INSERT INTO TEHILA.purchase(PURCHASE_ID, PURCHASE_DATE, DELLIVERY_DATE) VALUES(304, (TO_DATE('2025/07/25', 'yyyy/mm/dd')), (TO_DATE('2028/05/30', 'yyyy/mm/dd')));
INSERT INTO TEHILA.purchase(PURCHASE_ID, PURCHASE_DATE, DELLIVERY_DATE) VALUES(960, (TO_DATE('2027/11/23', 'yyyy/mm/dd')), (TO_DATE('2026/07/31', 'yyyy/mm/dd')));
INSERT INTO TEHILA.purchase(PURCHASE_ID, PURCHASE_DATE, DELLIVERY_DATE) VALUES(751, (TO_DATE('2026/01/11', 'yyyy/mm/dd')), (TO_DATE('2028/03/24', 'yyyy/mm/dd')));
INSERT INTO TEHILA.purchase(PURCHASE_ID, PURCHASE_DATE, DELLIVERY_DATE) VALUES(799, (TO_DATE('2028/07/10', 'yyyy/mm/dd')), (TO_DATE('2030/02/01', 'yyyy/mm/dd')));
INSERT INTO TEHILA.purchase(PURCHASE_ID, PURCHASE_DATE, DELLIVERY_DATE) VALUES(829, (TO_DATE('2026/01/12', 'yyyy/mm/dd')), (TO_DATE('2024/05/06', 'yyyy/mm/dd')));
INSERT INTO TEHILA.purchase(PURCHASE_ID, PURCHASE_DATE, DELLIVERY_DATE) VALUES(643, (TO_DATE('2029/09/11', 'yyyy/mm/dd')), (TO_DATE('2026/09/21', 'yyyy/mm/dd')));
INSERT INTO TEHILA.purchase(PURCHASE_ID, PURCHASE_DATE, DELLIVERY_DATE) VALUES(880, (TO_DATE('2028/08/07', 'yyyy/mm/dd')), (TO_DATE('2030/09/30', 'yyyy/mm/dd')));
INSERT INTO TEHILA.purchase(PURCHASE_ID, PURCHASE_DATE, DELLIVERY_DATE) VALUES(858, (TO_DATE('2029/11/19', 'yyyy/mm/dd')), (TO_DATE('2024/10/29', 'yyyy/mm/dd')));
```

SQL Window - New

SQL Output Statistics

```
insert into purchase (PURCHASE_ID, PURCHASE_DATE, DELLIVERY_DATE)
values (604, to_date('27-03-2025', 'dd-mm-yyyy'), to_date('16-10-2027', 'dd-mm-yyyy'));

insert into purchase (PURCHASE_ID, PURCHASE_DATE, DELLIVERY_DATE)
values (319, to_date('08-02-2025', 'dd-mm-yyyy'), to_date('02-05-2030', 'dd-mm-yyyy'));

insert into purchase (PURCHASE_ID, PURCHASE_DATE, DELLIVERY_DATE)
values (348, to_date('28-03-2025', 'dd-mm-yyyy'), to_date('27-01-2025', 'dd-mm-yyyy'));

insert into purchase (PURCHASE_ID, PURCHASE_DATE, DELLIVERY_DATE)
values (179, to_date('15-08-2025', 'dd-mm-yyyy'), to_date('14-08-2027', 'dd-mm-yyyy'));

insert into purchase (PURCHASE_ID, PURCHASE_DATE, DELLIVERY_DATE)
values (388, to_date('23-12-2026', 'dd-mm-yyyy'), to_date('07-05-2029', 'dd-mm-yyyy'));

insert into purchase (PURCHASE_ID, PURCHASE_DATE, DELLIVERY_DATE)
values (863, to_date('24-07-2025', 'dd-mm-yyyy'), to_date('14-03-2026', 'dd-mm-yyyy'));

insert into purchase (PURCHASE_ID, PURCHASE_DATE, DELLIVERY_DATE)
values (116, to_date('21-05-2029', 'dd-mm-yyyy'), to_date('23-12-2025', 'dd-mm-yyyy'));
```

SQL Output Statistics

select * from purchase

	PURCHASE_ID	PURCHASE_DATE	DELLIVERY_DATE
1	604	27/03/2025	16/10/2027
2	319	08/02/2025	02/05/2030
3	348	28/03/2025	27/01/2025
4	179	15/08/2025	14/08/2027
5	388	23/12/2026	07/05/2029
6	863	24/07/2025	14/03/2026
7	116	21/05/2029	23/12/2025
8	814	22/07/2029	05/08/2026
9	474	04/01/2024	01/06/2024
10	436	02/08/2028	21/10/2024
11	339	11/09/2025	17/01/2030
12	464	19/11/2028	06/02/2025
13	676	29/05/2025	20/01/2030

מיני פרויקט בבסיסי נתונים

הכנסת נתונים מקובץ:

E	D	C	B	A	
		IS_CLUB	CLIENT_N	CLIENT_I	1
		1	Rosanne	627	2
		1	Gwyneth	496	3
		0	Larry	818	4
		0	Joey	206	5
		0	Eliza	563	6
		1	Robin	126	7
		1	Gina	615	8
		0	Nastassja	551	9
		0	Angela	592	10
		0	Domingo	898	11
		1	Adam	700	12
		0	Simon	727	13
		0	Leon	115	14
		1	Anita	349	15
		1	Lauren	244	16
		0	Henry	487	17
		1	Famke	685	18
		0	Mia	462	19
		0	Celia	544	20
		0	Kevn	193	21
		0	Jack	951	22

Text Importer - Client.csv

Data from Textfile Data to Oracle

General

Owner: TEHILA Table: CLIENT ☐ Clear Table

Commit every...: 100 ☒ Overwrite duplicates ☐ Ignore duplicates

Initializing Script:

Finalizing Script:

Fields

Field1 CLIENT_ID -> CLIENT_ID
Field2 CLIENT_NAME -> CLIENT_NAME
Field3 IS_CLUB_MEMBER -> IS_CLUB_ME

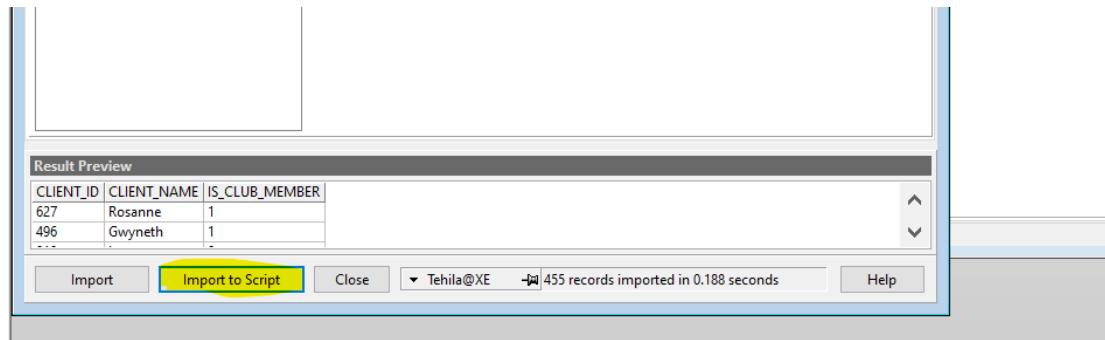
Field:
Fieldtype:
Create SQL:
SQL function:
additional Oracle processing, for example: substr(#, 1, 20)

Result Preview

CLIENT_ID	CLIENT_NAME	IS_CLUB_MEMBER
627	Rosanne	1
496	Gwyneth	1

Import Import to Script Close Tehila@XE Client.csv loaded, 8 KB Help

מיני פרויקט בבסיסי נתונים



```

SQL      Output  Statistics
insert into tehila.client (client_id, client_name, is_club_member)
values (627, 'Rosanne', 1);

insert into tehila.client (client_id, client_name, is_club_member)
values (496, 'Gwyneth', 1);

insert into tehila.client (client_id, client_name, is_club_member)
values (818, 'Larry', 0);

insert into tehila.client (client_id, client_name, is_club_member)
values (206, 'Joey', 0);

insert into tehila.client (client_id, client_name, is_club_member)
values (563, 'Eliza', 0);

insert into tehila.client (client_id, client_name, is_club_member)
values (126, 'Robin', 1);

insert into tehila.client (client_id, client_name, is_club_member)
values (615, 'Gina', 1);

insert into tehila.client (client_id, client_name, is_club_member)
values (551, 'Nastassja', 0);

insert into tehila.client (client_id, client_name, is_club_member)
values (592, 'Angela', 0);
    
```

SQL Window - SELECT * FROM CLIENT

SQL Output Statistics

SELECT * FROM CLIENT

	CLIENT_ID	CLIENT_NAME	IS_CLUB_MEMBER
1	627	Rosanne	1
2	496	Gwyneth	1
3	818	Larry	0
4	206	Joey	0
5	563	Eliza	0
6	126	Robin	1
7	615	Gina	1
8	551	Nastassja	0
9	592	Angela	0
10	898	Domingo	0
11	700	Adam	1
12	727	Simon	0
13	115	Leon	0
14	349	Anita	1
15	244	Lauren	1
16	487	Henry	0
17	685	Famke	1
18	462	Mia	0

מיני פרויקט בבסיסי נתונים

יצירת קובץ INSERT

שכולל כ- 10 שורות לכל אחת מהטבלאות כנדרש

```
insert into client (CLIENT_ID, CLIENT_NAME, IS_CLUB_MEMBER)
values (727, 'Simon', 0);

insert into client (CLIENT_ID, CLIENT_NAME, IS_CLUB_MEMBER)
values (115, 'Leon', 0);

insert into client (CLIENT_ID, CLIENT_NAME, IS_CLUB_MEMBER)
values (349, 'Anita', 1);

insert into client (CLIENT_ID, CLIENT_NAME, IS_CLUB_MEMBER)
values (244, 'Lauren', 1);

insert into CATEGORIES (CATEGORY_ID, CATEGORY_NAME)
values (100, 'irure ipsum do. ');

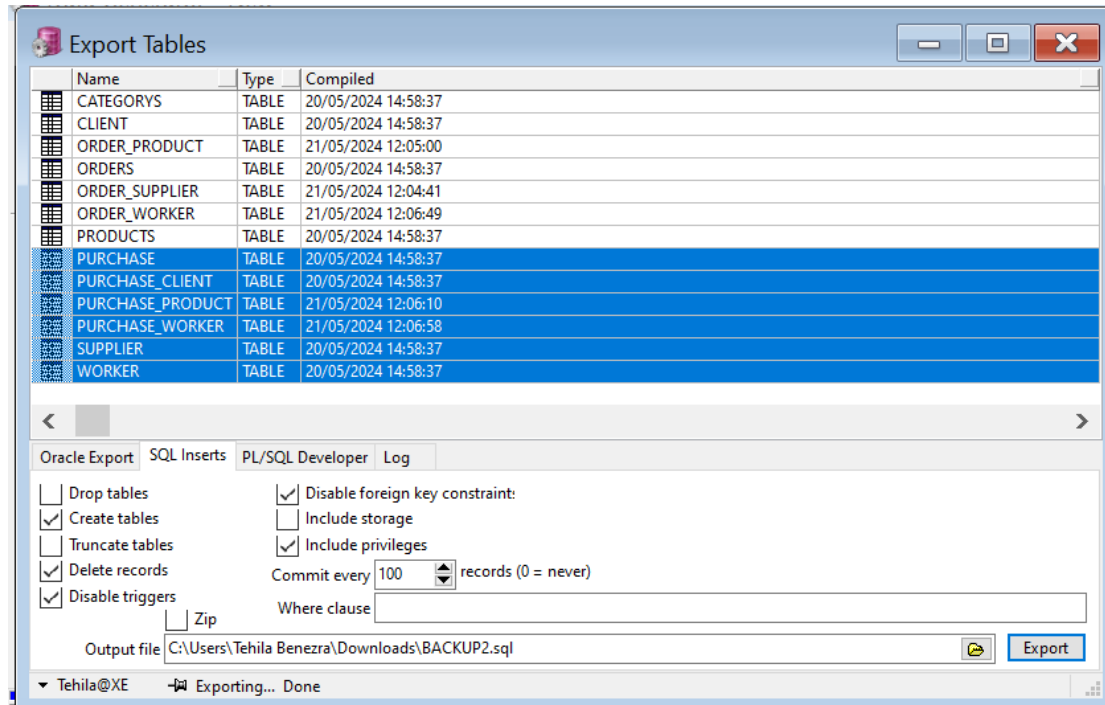
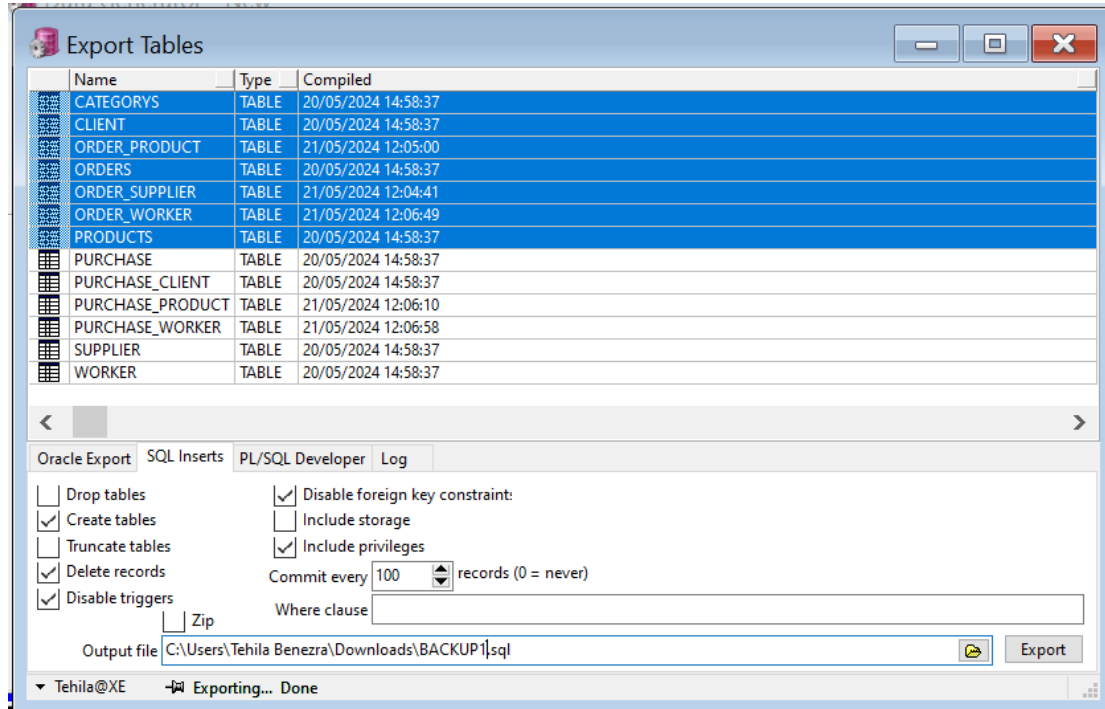
insert into CATEGORIES (CATEGORY_ID, CATEGORY_NAME)
values (101, 'rerum sit. ');

insert into CATEGORIES (CATEGORY_ID, CATEGORY_NAME)
values (102, 'officiis. ');
```

SQL	Output	Statistics
<pre>insert into ORDER_PRODUCT (PRODUCT_ID, ORDER_ID) values (111, 611); insert into ORDER_PRODUCT (PRODUCT_ID, ORDER_ID) values (117, 617); insert into ORDER_PRODUCT (PRODUCT_ID, ORDER_ID) values (120, 120); insert into ORDER_PRODUCT (PRODUCT_ID, ORDER_ID) values (124, 124); insert into ORDER_PRODUCT (PRODUCT_ID, ORDER_ID) values (126, 626); insert into ORDERS (ORDER_ID, ORDER_DATE, DELLIVERY_DATE) values (493, to_date('25-03-2025', 'dd-mm-yyyy'), to_date('03-06-2024', 'dd-mm-yyyy')); insert into ORDERS (ORDER_ID, ORDER_DATE, DELLIVERY_DATE) values (377, to_date('04-02-2026', 'dd-mm-yyyy'), to_date('23-05-2025', 'dd-mm-yyyy')); insert into ORDERS (ORDER_ID, ORDER_DATE, DELLIVERY_DATE) values (323, to_date('12-06-2026', 'dd-mm-yyyy'), to_date('10-04-2029', 'dd-mm-yyyy')); insert into ORDERS (ORDER_ID, ORDER_DATE, DELLIVERY_DATE) values (703, to_date('12-08-2025', 'dd-mm-yyyy'), to_date('24-12-2025', 'dd-mm-yyyy'));</pre>		
501:53 Tehila@XE SQL script saved successfully		

מיני פרויקט בבסיסי נתונים

ג'ירוי (הגיבוי בוצע ב2 חלקים כיוון שלא היה מקום)



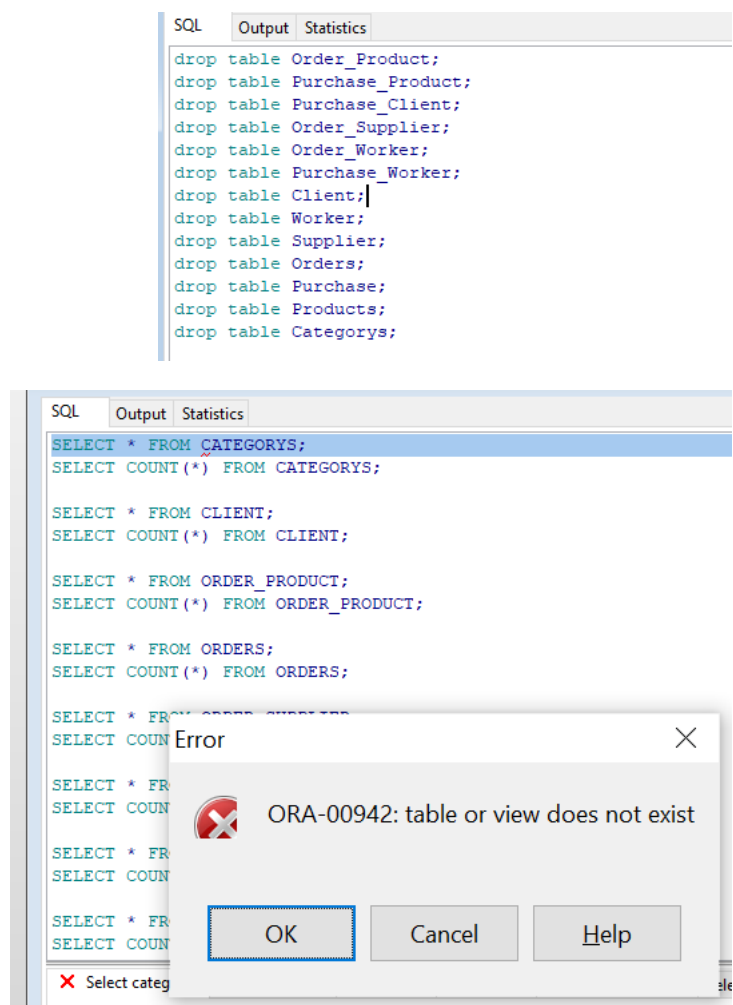
מיני פרויקט בבסיסי נתונים

```
BACKUP1.sql X
C: > Users > Tehila Benezra > Downloads > BACKUP1.sql
1 prompt PL/SQL Developer import file
2 prompt Created on יום 23 מאי 2024 by Tehila Benezra
3 set feedback off
4 set define off
5 prompt Creating CATEGORYS...
6 create table CATEGORYS
7 (
8     category_id    NUMBER(3) not null,
9     category_name  VARCHAR2(15) not null
10 )
11 ;
12 alter table CATEGORYS
13     add primary key (CATEGORY_ID);
14
15 prompt Creating CLIENT...
16 create table CLIENT
17 (
18     client_id      NUMBER(3) not null,
19     client_name    VARCHAR2(15) not null,
20     is_club_member NUMBER(1) not null
21 )
22 ;
23 alter table CLIENT
24     add primary key (CLIENT_ID);
25
26 prompt Creating ORDERS...
27 create table ORDERS
28 (
29     order_id       NUMBER(3) not null,
30     order_date     DATE not null,
```

```
C: > Users > Tehila Benezra > Downloads > BACKUP2.sql
1 prompt PL/SQL Developer import file
2 prompt Created on יום 23 מאי 2024 by Tehila Benezra
3 set feedback off
4 set define off
5 prompt Creating PURCHASE...
6 create table PURCHASE
7 (
8     purchase_id    NUMBER(3) not null,
9     purchase_date  DATE not null,
10    dellivery_date DATE not null
11 )
12 ;
13 alter table PURCHASE
14     add primary key (PURCHASE_ID);
15
16 prompt Creating PURCHASE_CLIENT...
17 create table PURCHASE_CLIENT
18 (
19     client_id      NUMBER(3) not null,
20     purchase_id    NUMBER(3) not null
21 )
22 ;
23 alter table PURCHASE_CLIENT
24     add primary key (CLIENT_ID, PURCHASE_ID);
25 alter table PURCHASE_CLIENT
26     add foreign key (CLIENT_ID)
27     references CLIENT (CLIENT_ID);
28 alter table PURCHASE_CLIENT
29     add foreign key (PURCHASE_ID)
30     references PURCHASE (PURCHASE ID);
```

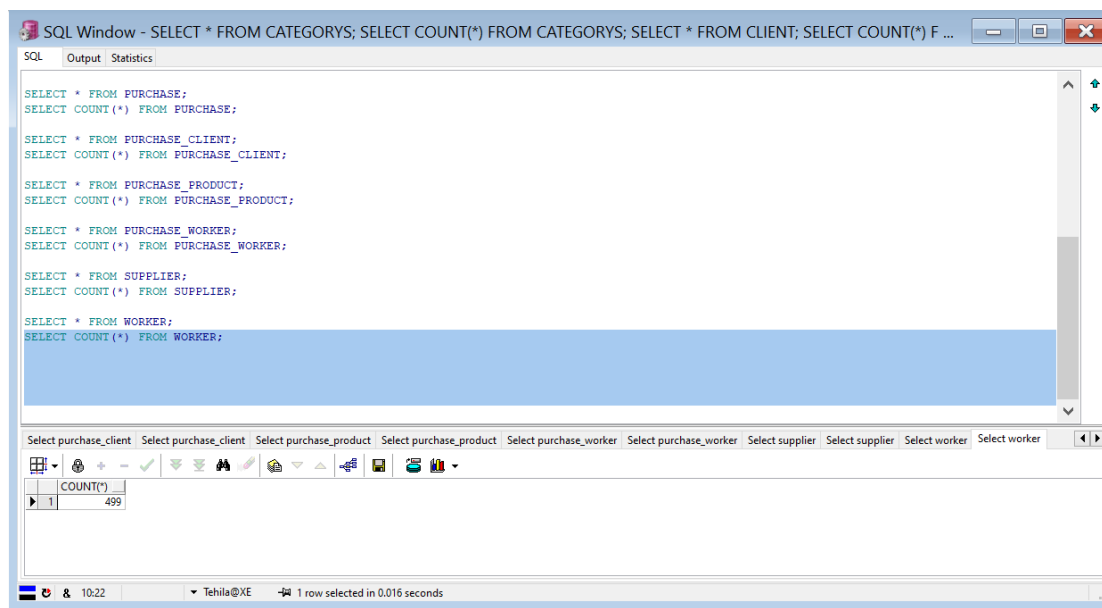
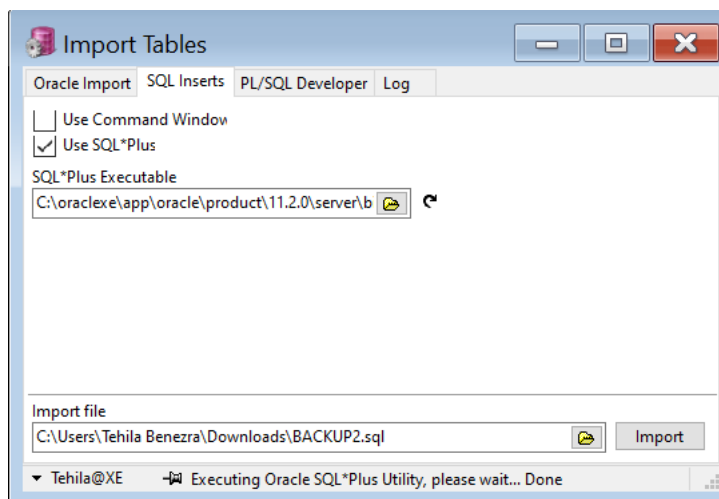
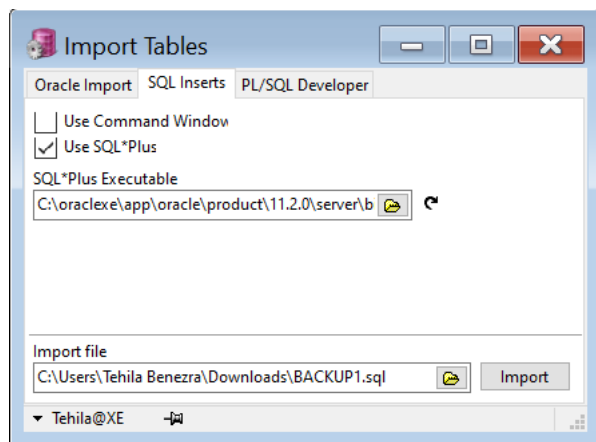
מיני פרויקט בבסיסי נתונים

מחיקת הפרויקט



מיני פרויקט בבסיסי נתונים

שחזור הפרויקט (כמובן עלינו לייבא את שני קבצי הגיבוי)



הפרויקט שוחזר בהצלחה.

פרק ב

Select query 1:

מחזיר את מספר הלקוחות שיש להם מועדון , ואת מספר
ההזמנות שרכשו

SQL	Output	Statistics
<pre>SELECT c.Client_Name, COUNT(pc.Purchase_Id) AS Total_Purchases FROM Client c INNER JOIN Purchase_Client pc ON c.Client_Id = pc.Client_Id WHERE c.Is_Club_Member = 1 GROUP BY c.Client_Name;</pre>		
CLIENT_NAME		TOTAL_PURCHASES
1	Lydia	1
2	Shannon	1
3	Rose	2
4	Joely	2
5	Jody	1
6	Ellen	1
7	Matt	1
8	Rascal	3
9	Ralph	1
10	Connie	2
11	Mandy	2
12	Gene	2
13	Willem	2
14	Samuel	1
15	Stanley	1
16	Shelby	2
17	Gavin	1
18	Olympia	3
19	Holland	1
20	Eugene	1
21	Johnnie	2
22	Jack	3
23	Merrilee	1
24	Frank	2
25	Gil	1
26	Don	2

Select query 2:

השאלתה מחזירה רשימה של קטגוריות מוצרים יחד עם המחיר הממוצע של המוצרים בכל קטגוריה, ממיון מהמחיר הממוצע הגבוה ביותר לנמוך ביותר. זה מאפשר השוואה קלה של מחירי המוצרים הממוצעים בקטגוריות שונות.

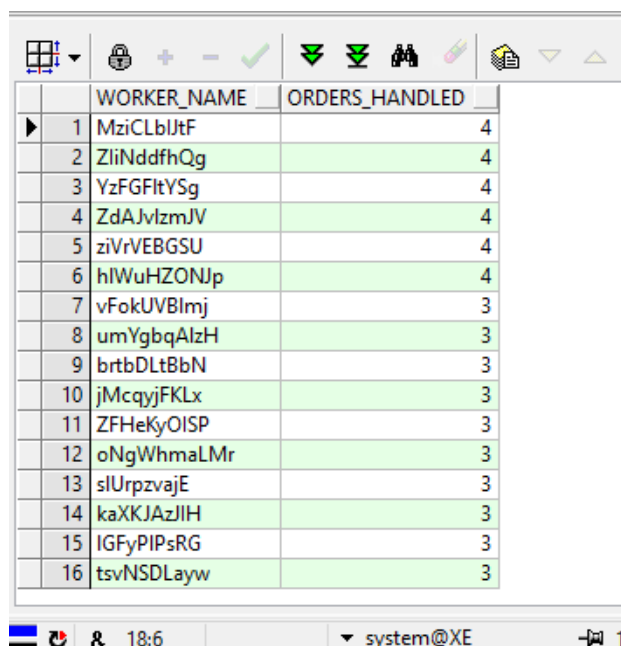
```
SQL Output Statistics
SELECT
  C.Category_Name,
  AVG(PR.Product_Price) AS Average_Price
FROM
  Categorys C
JOIN
  Products PR ON C.Category_Id = PR.Category_Id
GROUP BY
  C.Category_Name
ORDER BY
  Average_Price DESC;
```

	CATEGORY_NAME	AVERAGE_PRICE
1	irure ipsum do.	200
2	amet do	99
3	alias aut	99
4	minus at	99
5	laborum	97
6	dolores	96
7	sunt	95
8	ut culpa	94
9	non libero	94
10	quod	93
11	saepe in	93
12	earum	91
13	in qui	90
14	do	89.8571428571428
15	vero ut	88
16	proident ut	87
17	et vero qui	85
18	amet aute	84
19	nostrud	83.5

Select query 3:

שאלתה לרשימת עובדים שטיפלו ביותר הזמנות ממספר ההזמנות הממוצע שטופלו על ידי כל העובדים, מסודרים לפי מספר ההזמנות שטופלו בסדר יורד:

```
SELECT
    w.Worker_Name,
    COUNT(ow.Order_Id) AS Orders_Handled
FROM
    Worker w
JOIN
    Order_Worker ow ON w.Worker_Id = ow.Worker_Id
GROUP BY
    w.Worker_Name
HAVING
    COUNT(ow.Order_Id) > (
        SELECT AVG(Order_Count)
        FROM (
            SELECT COUNT(Order_Id) AS Order_Count
            FROM Order_Worker
            GROUP BY Worker_Id
        )
    )
ORDER BY
    Orders_Handled DESC;
```



	WORKER_NAME	ORDERS_HANDLED
1	MziCLbJtF	4
2	ZliNddfhQg	4
3	YzFGFitYSg	4
4	ZdAJvlzmJV	4
5	ziVrVEBGsu	4
6	hIWuHZONJp	4
7	vFokUVBlmj	3
8	umYgbqAlzH	3
9	brtbDLtBbN	3
10	jMcqyjFKLx	3
11	ZFHeKyOISP	3
12	oNgWhmaLMr	3
13	slUrpzvajE	3
14	kaXKJAzJIH	3
15	IGFyPIPsRG	3
16	tsvNSDLayw	3

Select query 4:

השאלתה מחזירה רשימה של עובדים יחד עם ספירת ההזמנות להן
הוקצו

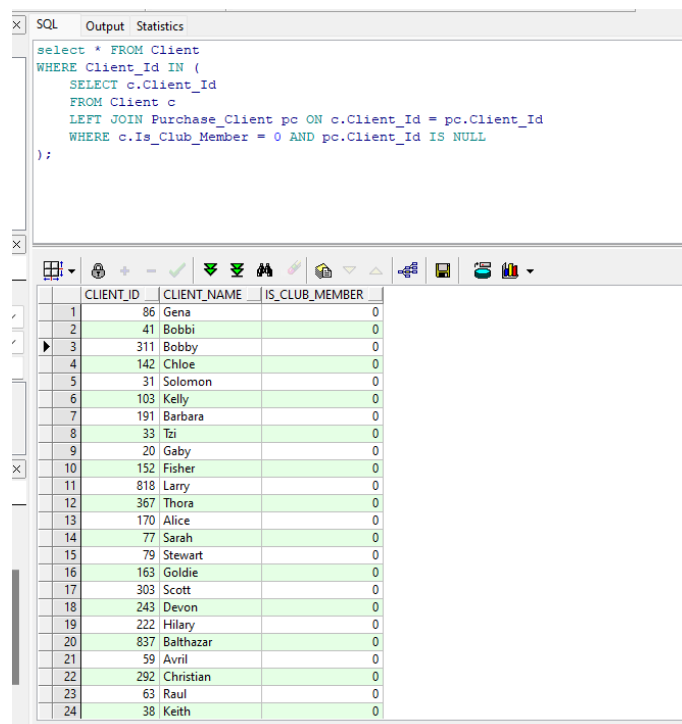
```
SQL Output Statistics
SELECT w.Worker_Name, COUNT(ow.Order_Id) AS Assigned_Order_Count
FROM Worker w
LEFT JOIN Order_Worker ow ON w.Worker_Id = ow.Worker_Id
GROUP BY w.Worker_Name;
```

	WORKER_NAME	ASSIGNED_ORDER_COUNT
1	puxetLwKil	1
2	qHyXvQubmi	0
3	LMHneGcjSs	0
4	qBDWviOnPd	1
5	oTFsXBdyST	2
6	RzzphUKGOv	1
7	IGFyPIPsRG	3
8	HxnwyDaczK	1
9	rWiNIKuDwj	2
10	KgZbtaNcPG	2
11	DTHJVLijsL	0
12	aALtGlqGdO	0
13	tsvNSDLayw	3
14	qsrpJdGOqW	0
15	oHWpEQZeSd	2
16	SoEJlkEctp	2
17	xSxBrgiPFY	1

Delete query 1:

יצרנו שאילתת מחיקה , מוחקת את הלקוחות שלא חברי מועדון
וביצעו רכישות בחנות .

נתון במידע לפני המחיקה:

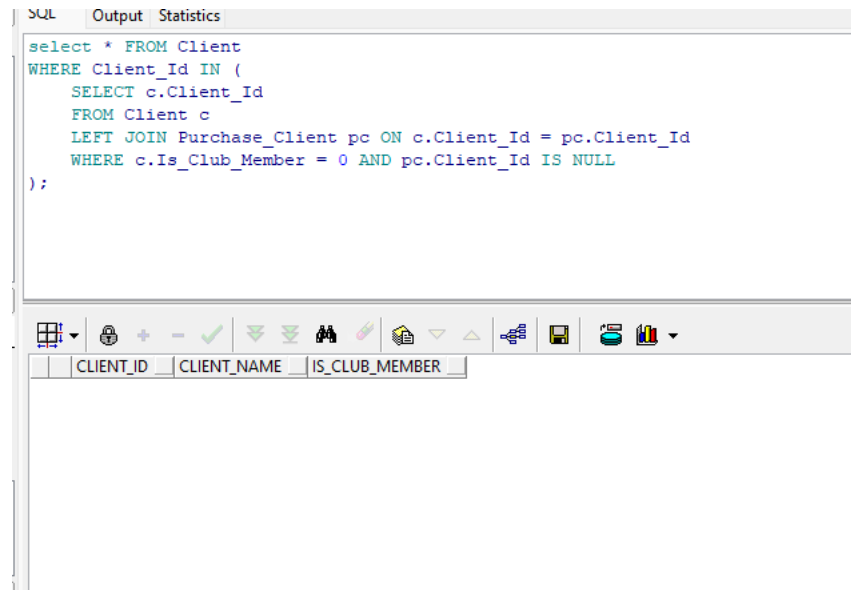


The screenshot shows the SQL Server Enterprise Manager interface. The top pane displays a SQL query designed to delete clients who are not club members but have made purchases. The bottom pane shows the results of the query, which lists 24 clients with their IDs, names, and club membership status (all 0).

```
select * FROM Client
WHERE Client_Id IN (
    SELECT c.Client_Id
    FROM Client c
    LEFT JOIN Purchase_Client pc ON c.Client_Id = pc.Client_Id
    WHERE c.Is_Club_Member = 0 AND pc.Client_Id IS NULL
);
```

	CLIENT_ID	CLIENT_NAME	IS_CLUB_MEMBER
1	86	Gena	0
2	41	Bobbi	0
3	311	Bobby	0
4	142	Chloe	0
5	31	Solomon	0
6	103	Kelly	0
7	191	Barbara	0
8	33	Tzi	0
9	20	Gaby	0
10	152	Fisher	0
11	818	Larry	0
12	367	Thora	0
13	170	Alice	0
14	77	Sarah	0
15	79	Stewart	0
16	163	Goldie	0
17	303	Scott	0
18	243	Devon	0
19	222	Hilary	0
20	837	Balthazar	0
21	59	Avril	0
22	292	Christian	0
23	63	Raul	0
24	38	Keith	0

נעשה צילום מסך אחרי המחיקה :

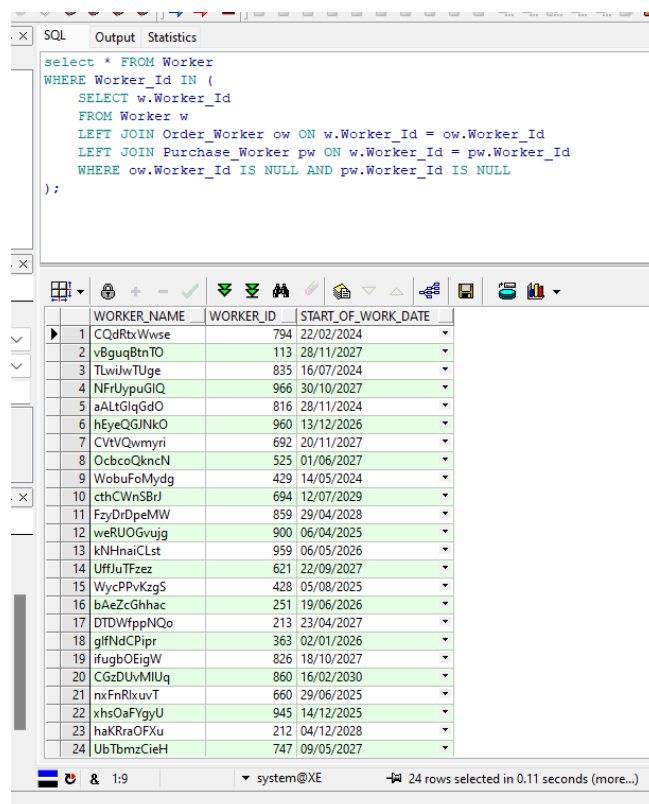


Delete query 2:

יצרנו שאילתת מחיקה נוספת .

שמוחקת את כל העובדים שמעולם לא הוקצו להזמנות ורכישות

אלא הנתונים לפני המחיקה :



לאחר המחיקה כך נראה בסיס הנתונים שאליו ניסינו לגשת :

```

SQL Output Statistics
select * FROM Worker
WHERE Worker_Id IN (
  SELECT w.Worker_Id
  FROM Worker w
  LEFT JOIN Order_Worker ow ON w.Worker_Id = ow.Worker_Id
  LEFT JOIN Purchase_Worker pw ON w.Worker_Id = pw.Worker_Id
  WHERE ow.Worker_Id IS NULL AND pw.Worker_Id IS NULL
);

/*delete FROM Worker
WHERE Worker_Id IN (
  SELECT v.Worker_Id
  FROM Worker v
  LEFT JOIN Order_Worker ow ON v.Worker_Id = ow.Worker_Id
  LEFT JOIN Purchase_Worker pw ON v.Worker_Id = pw.Worker_Id
  WHERE ow.Worker_Id IS NULL AND pw.Worker_Id IS NULL
);*/

```

WORKER_NAME	WORKER_ID	START_OF_WORK_DATE
-------------	-----------	--------------------

Update query 1

שאלתה שמעלה את המחיר מוצר בקטגוריה מסוימת ב10 אחוז
לפני:

```

SQL Output Statistics
select* from Products
SET Product_Price = Product_Price * 1.10
WHERE Category_Id = (
  SELECT Category_Id
  FROM Categorys
  WHERE Category_Name = 'irure ipsum do.'
);

```

	PRODUCT_ID	PRODUCT_NAME	QUANTITY	PRODUCT PRICE	CATEGORY_ID
▶ 1	100	Alden Systems	200	200	100
2	200	Manhattan Assoc	200	200	100
3	300	Street Glow	200	200	100
4	400	General Motors	200	200	100
5	500	Safeway	200	200	100
6	600	Novartis	200	200	100

אחרי:

SQL Output Statistics					
<pre>--UPDATE Products select * from Products --SET Product_Price = Product_Price * 1.10 WHERE Category_Id = (SELECT Category_Id FROM Categorys WHERE Category_Name = 'irure ipsum do.');</pre>					
	PRODUCT_ID	PRODUCT_NAME	QUANTITY	PRODUCT_PRICE	CATEGORY_ID
1	100	Alden Systems	200	220	100
2	200	Manhattan Assoc	200	220	100
3	300	Street Glow	200	220	100
4	400	General Motors	200	220	100
5	500	Safeway	200	220	100
6	600	Novartis	200	220	100

Update query 2:

שאלת עדכון שמוסיפה לתאריך משלוח 7 ימים

לפני

```
select * from Orders
WHERE Order_Id IN (
    SELECT os.Order_Id
    FROM Order_Supplier os
    JOIN Supplier s ON os.Supplier_Id = s.Supplier_Id
    WHERE s.Region = 'Boucherville'
);
```

	ORDER_ID	ORDER_DATE	DELLIVERY_DATE
1	996	12/08/2026	09/10/2025
2	362	13/02/2025	15/12/2027
3	432	11/10/2028	13/10/2029

אחרי

```
select *
from Orders
WHERE Order_Id IN (
    SELECT os.Order_Id
    FROM Order_Supplier os
    JOIN Supplier s ON os.Supplier_Id = s.Supplier_Id
    WHERE s.Region = 'Boucherville'
);
```

	ORDER_ID	ORDER_DATE	DELLIVERY_DATE
1	996	12/08/2026	16/10/2025
2	362	13/02/2025	22/12/2027
3	432	11/10/2028	20/10/2029

השאלתה

```
UPDATE Orders
SET Dellivery_Date = Dellivery_Date + 7
WHERE Order_Id IN (
    SELECT os.Order_Id
    FROM Order_Supplier os
    JOIN Supplier s ON os.Supplier_Id = s.Supplier_Id
    WHERE s.Region = 'Boucherville'
```

Queries with parameters:

1.query with data parameter

שאלתה שמחזירה את כל הנתונים של ההזמנה מתאריך מסוים או אחריו

לדוג: כאן קיבלנו את כל ההזמנות שבוצעו מתאריך 12.06.2026

The screenshot displays the SQL Server Enterprise Manager interface. The SQL query editor shows a query that selects order details from the 'Orders' table, joined with 'Purchase_Client' and 'Client' tables. The query filters for orders on or after the date specified in the '&order_date' parameter. The 'Variables' dialog box is open, showing the 'order_date' variable set to '12/06/2026'. Below the query editor, the results grid shows 18 rows of data with columns: ORDER_ID, ORDER_DATE, DELIVERY_DATE, and CLIENT_NAME.

```
SELECT
    O.Order_Id,
    O.Order_Date,
    O.Dellivery_Date,
    C.Client_Name
FROM
    Orders O
JOIN
    Purchase_Client PC ON O.Order_Id = PC.Purchase_Id
JOIN
    Client C ON PC.Client_Id = C.Client_Id
WHERE
    O.Order_Date >= TO_DATE('&order_date', 'dd/mm/yyyy');
```

	ORDER_ID	ORDER_DATE	DELLIVERY_DATE	CLIENT_NAME
1	814	13/10/2030	01/03/2029	Rita
2	611	01/12/2030	18/05/2027	Lila
3	814	13/10/2030	01/03/2029	Sinead
4	611	01/12/2030	18/05/2027	Charles
5	814	13/10/2030	01/03/2029	Shelby
6	611	01/12/2030	18/05/2027	Lin
7	604	10/10/2030	06/01/2029	Mekhi
8	436	29/06/2030	27/11/2030	Miki
9	474	09/09/2030	01/08/2029	Corey
10	474	09/09/2030	01/08/2029	Seth
11	604	10/10/2030	06/01/2029	Stevie
12	101	06/07/2027	03/07/2024	Kurtwood
13	104	19/10/2028	07/11/2026	Juliana
14	105	13/11/2028	24/04/2024	Lili
15	106	13/09/2026	06/04/2026	Philip
16	604	10/10/2030	06/01/2029	Ralph
17	110	23/08/2027	11/11/2030	Rowan
18	111	08/05/2027	20/09/2024	Holland

2.query with date parameter

השאלתה מחזירה נתונים של עובד שהחל לעבוד בטווח תאריכים מסוים.
לדוג כאן הוא החזיר נתונים של כל העובדים שתאריך העבודה ההתחלתי
שלהם הוא בין 4.5.28 ל 4.5.29

The screenshot shows the SQL Developer interface. The main window displays a SQL query titled 'parameter2.sql'. The query selects worker information based on a date range defined by two variables, 'start_date' and 'end_date'. A 'Variables' dialog box is open, showing the values assigned to these variables: 'start_date' is '04/05/2028' and 'end_date' is '04/05/2029'. Below the query editor, the results of the query are displayed in a table.

```
SQL Window - parameter2.sql
SQL Output Statistics
SELECT
  W.Worker_Id,
  W.Worker_Name,
  W.Start_of_Work_date
FROM
  Worker W
WHERE
  W.Start_of_Work_date >= TO_DATE('&start_date', 'dd/mm/yyyy')
  AND W.Start_of_Work_date <= TO_DATE('&end_date', 'dd/mm/yyyy');
```

Variables

Name	Value
start_date	04/05/2028
end_date	04/05/2029

WORKER_ID	WORKER_NAME	START_OF_WORK_DATE
1	131 VVZPsTNsdu	09/02/2029
2	807 DfaqtdNerj	06/08/2028
3	105 QmWenKDtjy	29/06/2028
4	744 LEolmByKar	24/10/2028
5	209 qBDWviOnPd	19/06/2028
6	666 sGjuiFIfkw	07/05/2028
7	518 jnguVFOEtX	13/10/2028
8	188 igFPQwpXZJ	03/05/2029
9	514 vscBzCjJhH	20/10/2028
10	761 IGFyPIPsRG	07/07/2028
11	340 VlmXfövlkT	08/12/2028

3.query with hint parameter

The screenshot shows a SQL Window titled "parameter3.sql" with a query that uses a hint parameter. The query is:

```
SELECT
  C.client_id,
  C.client_name,
  C.is_club_member
FROM
  client C
WHERE
  C.Client_Id=&<name="clientID" hint="client id value between 0-999">;
```

A "Variables" dialog box is open, showing a table with two columns: "Name" and "Value". The table contains one row: "clientID" with the value "627". The dialog box has "OK", "Cancel", and "Clear" buttons. Below the dialog box, the text "client id value between 0-999" is visible.

	CLIENT_ID	CLIENT_NAME	IS_CLUB_MEMBER
1	627	Rosanne	1

4.query with name parameter

The screenshot shows a SQL Window titled "parameter4.sql" with a query that uses a name parameter. The query is:

```
SELECT
  O.Order_Id,
  O.Order_Date,
  O.Delivery_Date
FROM
  Orders O
JOIN
  Purchase_Client PC ON O.Order_Id = PC.Purchase_Id
JOIN
  Client C ON PC.Client_Id = C.Client_Id
WHERE
  C.Client_Name = '&client_name';
```

A "Variables" dialog box is open, showing a table with two columns: "Name" and "Value". The table contains one row: "client_name" with the value "Joey". The dialog box has "OK", "Cancel", and "Clear" buttons.

	ORDER_ID	ORDER_DATE	DELIVERY_DATE
1	206	22/09/2029	07/10/2029
2	106	13/09/2026	06/04/2026
3	133	09/01/2028	13/06/2025

5.query with list parameter

SQL Window - parameter5.sql

```

select
c.category_name,
c.category_id,
pr.product_id,
pr.product_name
from
categories c
join
products pr on pr.category_id = c.category_id
where
c.category_id=&name="category_name"
list="select category_id,category_name from categories order by category_name"
description="yes" restricted="yes" >

```

Variables

Name	Value
category_name	iusto

OK Cancel Clear

	CATEGORY_NAME	CATEGORY_ID	PRODUCT_ID	PRODUCT_NAME
1	iusto	130	130	Procter & Gambl
2	iusto	130	230	Best Buy Co.
3	iusto	130	330	GCI
4	iusto	130	430	Client Network
5	iusto	130	530	Newton Interact
6	iusto	130	630	Viacom

Constraints

1.alter table –CHECK

```
SQL Output Statistics
-- Adding the CHECK constraint
ALTER TABLE Products
ADD CONSTRAINT CHK_Quantity2 CHECK (Quantity >= 0);

-- Attempting to insert a product with Quantity = -1 (this should fail)
INSERT INTO Products (Product_Id, Product_Name, Quantity, Product_Price, Category_Id)
VALUES (101, 'Test Product', -1, 10, 1);
```

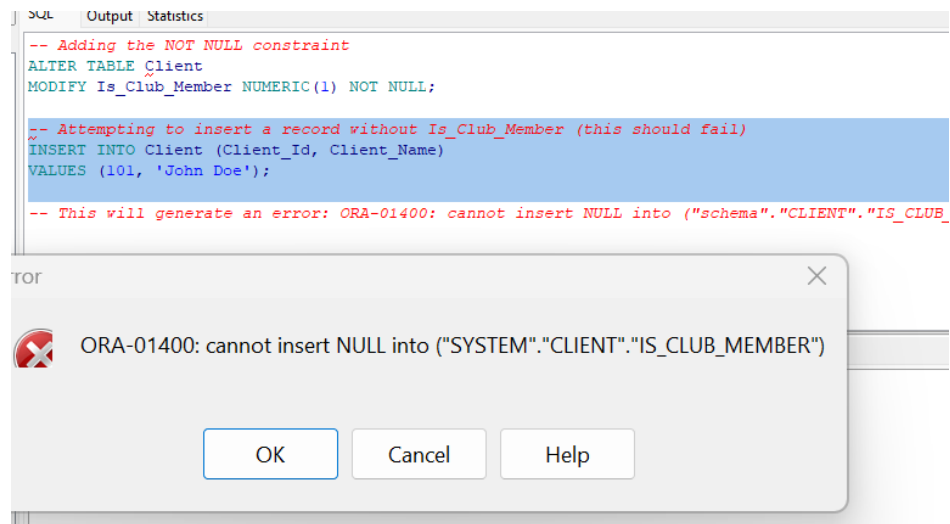
2.alter table-DEFAULT

```
SQL Output Statistics
-- Adding the DEFAULT constraint
ALTER TABLE Supplier
MODIFY Region VARCHAR(15) DEFAULT 'Unknown';

-- Inserting a supplier without specifying the Region (Region should default to 'Unknown')
INSERT INTO Supplier (Supplier_Id, Supplier_Name)
VALUES (226, 'ABC Suppliers');

-- Verifying the insertion
SELECT * FROM Supplier WHERE Supplier_Id = 226;
```

הוא לא נותן לי להוסיף לקוח בגלל שלא הכנסנו ערך ואין אפשרות של הוספת ערך NULL , הגדרנו בטבלאות שאין אפשרות הזנת ערך NULL

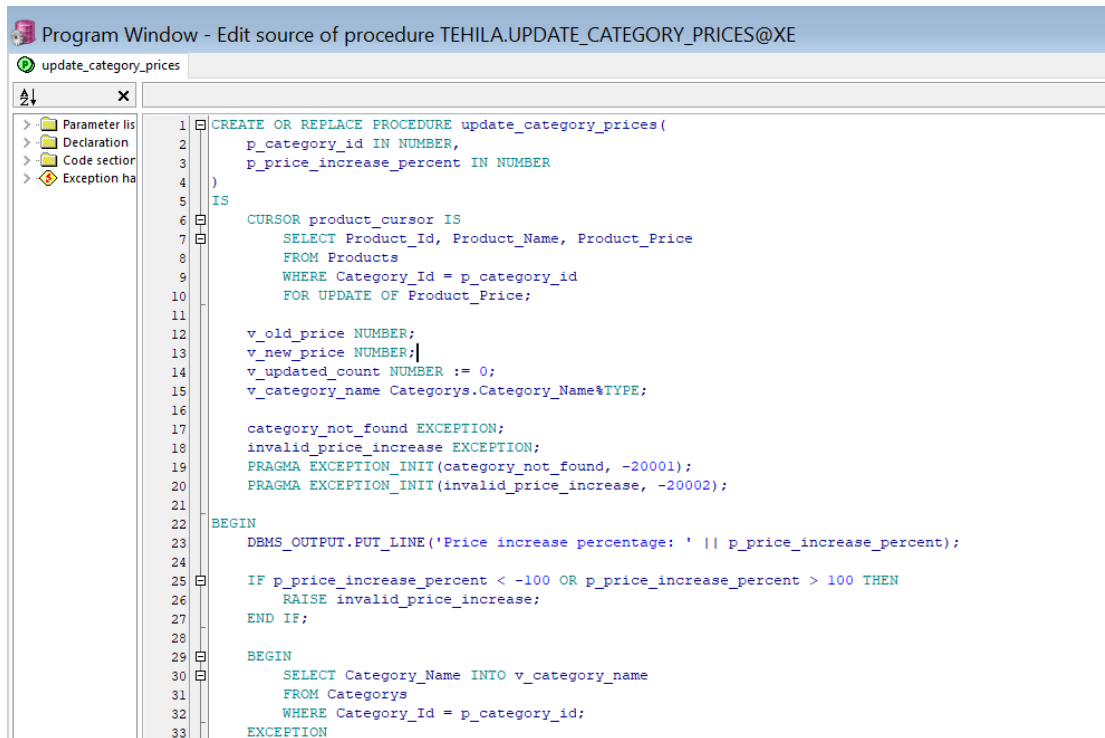


שלב 3

פרוצדורה #1

פרוצדורה לעדכון מחירים של מוצרים מאותה קטגוריה

בפרוצדורה נעשה שימוש בexplicit cursor, רשומות, תנאים, חריגות, פקודות DML וולאות.



```
Program Window - Edit source of procedure TEHILA.UPDATE_CATEGORY_PRICES@XE
update_category_prices

1 CREATE OR REPLACE PROCEDURE update_category_prices(
2   p_category_id IN NUMBER,
3   p_price_increase_percent IN NUMBER
4 )
5 IS
6   CURSOR product_cursor IS
7     SELECT Product_Id, Product_Name, Product_Price
8     FROM Products
9     WHERE Category_Id = p_category_id
10    FOR UPDATE OF Product_Price;
11
12   v_old_price NUMBER;
13   v_new_price NUMBER;
14   v_updated_count NUMBER := 0;
15   v_category_name Categories.Category_Name%TYPE;
16
17   category_not_found EXCEPTION;
18   invalid_price_increase EXCEPTION;
19   PRAGMA EXCEPTION_INIT(category_not_found, -20001);
20   PRAGMA EXCEPTION_INIT(invalid_price_increase, -20002);
21
22 BEGIN
23   DBMS_OUTPUT.PUT_LINE('Price increase percentage: ' || p_price_increase_percent);
24
25   IF p_price_increase_percent < -100 OR p_price_increase_percent > 100 THEN
26     RAISE invalid_price_increase;
27   END IF;
28
29   BEGIN
30     SELECT Category_Name INTO v_category_name
31     FROM Categories
32     WHERE Category_Id = p_category_id;
33   EXCEPTION
```

(הקוד המלא מצורף כמובן כקובץ נפרד)

בסיס הנתונים לפני הרצת הפרוצדורה:

SQL Window - commit; select * from products where category_id=121;

SQL Output Statistics

```
commit;
select * from products
where category_id=121;
```

Commit Select products

	PRODUCT_ID	PRODUCT_NAME	QUANTITY	PRODUCT_PRICE	CATEGORY_ID
1	121	Hat World	21	55	121
2	221	Capella Educati	21	55	121
3	321	Gentra Systems	21	55	121
4	421	Joseph Sheairs	21	55	121
5	521	American Expres	21	55	121
6	621	Benecon Group	21	55	121

הרצת הפרוצדורה:

Test Window - Script for procedure UPDATE_CATEGORY_PRICES@XE

Test script DBMS Output Statistics Profiler Trace

```
1 BEGIN
2   update_category_prices(121, 10);
3 END;
4 |
```

	Variable	Type	Value
<input type="checkbox"/>	p_category_id	Float	121
<input type="checkbox"/>	p_price_increase_percent	Float	10
*			

4:1 Tehila@XE Executed in 0 seconds

ניתן לראות את הפלט של הפרוצדורה:

```

Test Window - Script for procedure UPDATE_CATEGORY_PRICES@XE
Test script DBMS Output Statistics Profiler Trace
Clear Buffer size 10000 Enabled

Product: Capella Educati
Old price: 55
Multiplier: 1.10
New price before rounding: 60.50
New price after rounding: 60.5
Updated product: Capella Educati, Old price: 55, New price: 60.5
-----
Product: Gentra Systems
Old price: 55
Multiplier: 1.10
New price before rounding: 60.50
New price after rounding: 60.5
Updated product: Gentra Systems, Old price: 55, New price: 60.5
-----
Product: Joseph Sheairs
Old price: 55
Multiplier: 1.10
New price before rounding: 60.50
New price after rounding: 60.5
Updated product: Joseph Sheairs, Old price: 55, New price: 60.5
-----
Product: American Expres
Old price: 55
Multiplier: 1.10
New price before rounding: 60.50
New price after rounding: 60.5
Updated product: American Expres, Old price: 55, New price: 60.5
-----
Product: Benecon Group
Old price: 55
Multiplier: 1.10
New price before rounding: 60.50
New price after rounding: 60.5
Updated product: Benecon Group, Old price: 55, New price: 60.5
-----
Total updated products in category sint
: 6

```

בדיקת בסיס הנתונים לאחר ההרצה:

ניתן לראות שהמחירים אכן עודכנו (אצלנו המחיר הוא מטיפוס INT):

```

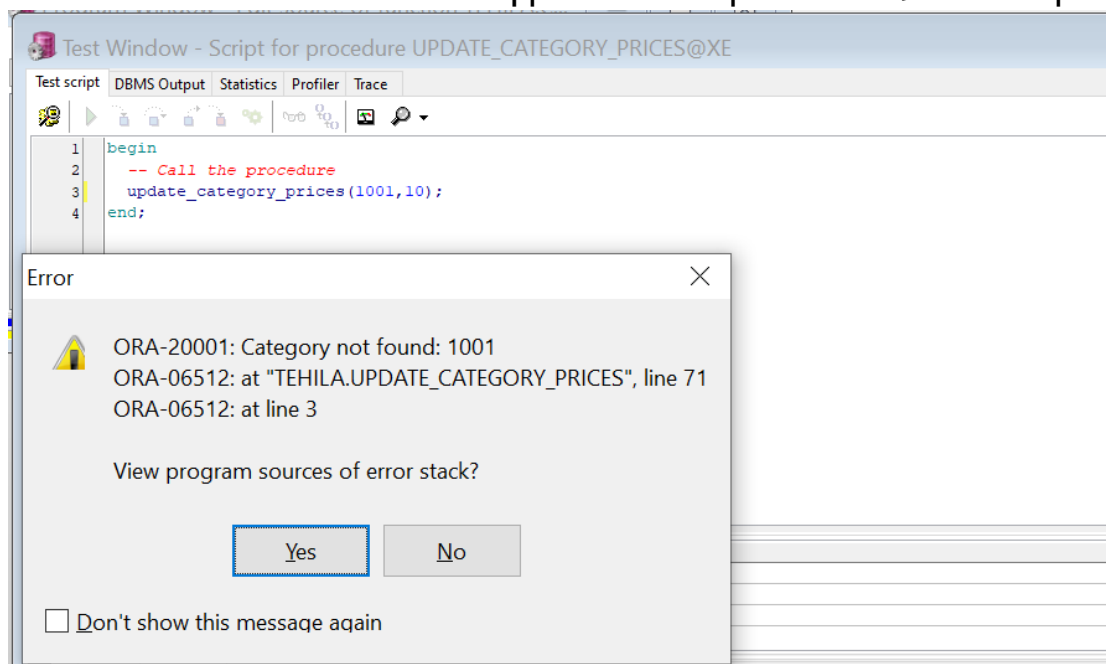
SQL Window - commit; select * from products where category_id=121;
SQL Output Statistics
commit;
select * from products
where category_id=121;

```

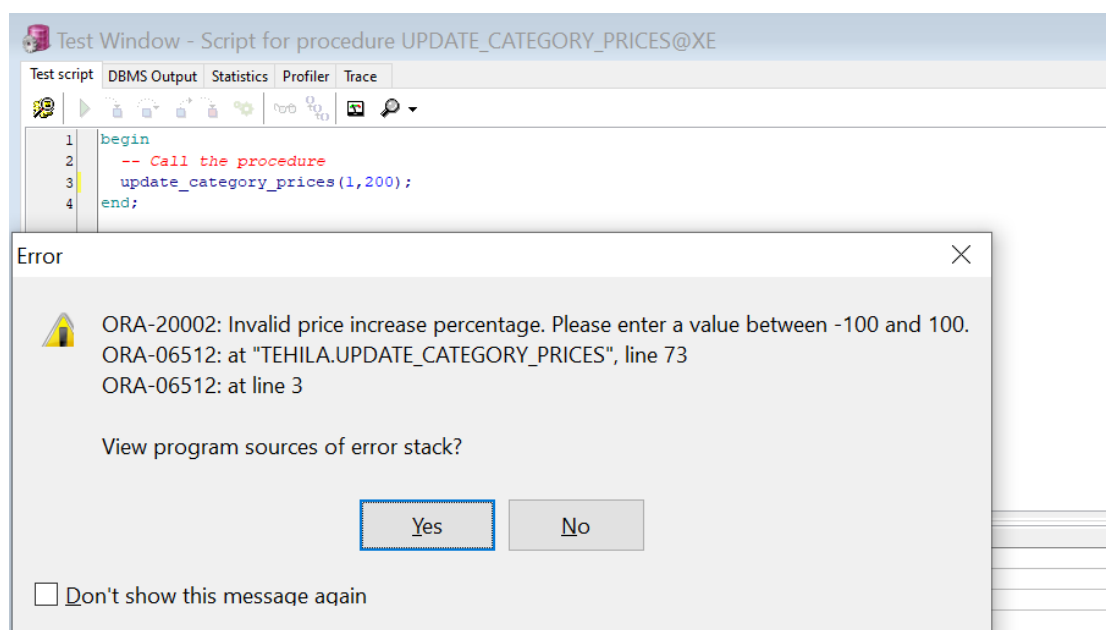
	PRODUCT_ID	PRODUCT_NAME	QUANTITY	PRODUCT_PRICE	CATEGORY_ID
1	121	Hat World	21	61	121
2	221	Capella Educati	21	61	121
3	321	Gentra Systems	21	61	121
4	421	Joseph Sheairs	21	61	121
5	521	American Expres	21	61	121
6	621	Benecon Group	21	61	121

טיפול בשגיאות:

זריקת חריגה עבור מספר קטגוריה לא תקין:



זריקת חריגה עבור ערך אחוז לא תקין:



פרוצדורה #2

Analyze_supplier_order

הפרוצדורה מנתחת הזמנות של ספק מסוים על פי מזהה הספק שמועבר כפרמטר (supplier_id).

היא מחשבת מספר הזמנות, סך המוצרים שהוזמנו, המוצר המוזמן ביותר, ותאריך ההזמנה האחרון.

יש שימוש ב-Explicit Cursor בשם c_orders שמוגדר בתחילת הפרוצדורה ומשמש לאיסוף נתוני ההזמנות.

יש מספר הסתעפויות ב-IF statements,

למשל:

- בדיקה אם זו הזמנה חדשה
- בדיקה אם זו ההזמנה הראשונה
- בדיקה אם המוצר הנוכחי הוא המוזמן ביותר

יש שימוש בלולאת FOR לעבור על תוצאות ה-Cursor:

FOR order_rec IN c_orders LOOP

וכמובן יש שימוש ב-Exception handling בשני מקומות:

- בבזק הפנימי שמנסה לקבל את שם הספק
- בזק הפרוצדורה לתפוס כל שגיאה אחרת שעלולה להתרחש

בזק הפרוצדורה משתמשת ב-(v_product_counts) associative array לספירת המוצרים.

זן היא משתמשת ב-DBMS_OUTPUT.PUT_LINE להדפסת התוצאות.

הפרוצדורה: (הקובץ המלא מצורף בנפרד)

```
1 CREATE OR REPLACE PROCEDURE analyze_supplier_orders(  
2   p_supplier_id IN Supplier.Supplier_Id%TYPE  
3 )  
4 IS  
5   v_supplier_name Supplier.Supplier_Name%TYPE;  
6   v_order_count NUMBER := 0;  
7   v_total_products NUMBER := 0;  
8   v_most_ordered_product Products.Product_Name%TYPE;  
9   v_most_ordered_quantity NUMBER := 0;  
10  v_last_order_date Orders.Order_Date%TYPE;  
11  v_current_order_id Orders.Order_Id%TYPE := -1;  
12  CURSOR c_orders IS  
13    SELECT o.Order_Id, o.Order_Date, p.Product_Id, p.Product_Name  
14    FROM Orders o  
15    JOIN Order_Supplier os ON o.Order_Id = os.Order_Id  
16    JOIN Order_Product op ON o.Order_Id = op.Order_Id  
17    JOIN Products p ON op.Product_Id = p.Product_Id  
18    WHERE os.Supplier_Id = p_supplier_id  
19    ORDER BY o.Order_Date DESC;  
20  TYPE product_count_type IS TABLE OF NUMBER INDEX BY PLS_INTEGER;  
21  v_product_counts product_count_type;  
22 BEGIN  
23   -- Get supplier name  
24   BEGIN  
25     SELECT Supplier_Name INTO v_supplier_name  
26     FROM Supplier  
27     WHERE Supplier_Id = p_supplier_id;  
28  
29     DBMS_OUTPUT.PUT_LINE('Analyzing orders for supplier: ' || v_supplier_name);  
30   EXCEPTION  
31     WHEN NO_DATA_FOUND THEN  
32       DBMS_OUTPUT.PUT_LINE('No supplier found with ID: ' || p_supplier_id);  
33       RETURN;  
34   END;  
35  
36   -- Analyze orders  
37   FOR order_rec IN c_orders LOOP  
38     IF order_rec.Order_Id != v_current_order_id THEN  
39       v_order_count := v_order_count + 1;  
40       v_current_order_id := order_rec.Order_Id;  
41  
42       -- Set last order date (will be the first in the cursor due to DESC order)  
43       IF v_order_count = 1 THEN  
44         v_last_order_date := order_rec.Order_Date;  
45       END IF;  
46     END IF;  
47   END LOOP;  
48  
49   -- Calculate total products and most ordered product  
50   FORALL v_product_counts  
51     v_product_counts(v_order_count) := v_total_products;  
52   FORALL v_product_counts  
53     v_product_counts(v_order_count) := v_most_ordered_product;  
54   v_total_products := v_total_products + 1;  
55   v_most_ordered_quantity := v_most_ordered_quantity + 1;  
56  
57   -- Update most ordered product and quantity  
58   IF v_most_ordered_quantity > 1 THEN  
59     v_most_ordered_product := v_most_ordered_product;  
60   ELSE  
61     v_most_ordered_product := v_most_ordered_product;  
62   END IF;  
63  
64   -- Update last order date  
65   v_last_order_date := v_last_order_date;  
66  
67   -- Return results  
68   RETURN;  
69 END;
```

nila@XE Compiled successfully

הרצת הפרוצדורה:

```
1 begin  
2   -- Call the procedure  
3   analyze_supplier_orders(101);  
4 end;
```

הפלט שקיבלנו:

Test script DBMS Output Statistics Profiler Trace

Clear Buffer size 10000 ☒ Enabled

```

Analyzing orders for supplier: Steve
Analysis for supplier: Steve
Total orders: 2
Total products ordered: 11
Most ordered product: Montpelier Plas (Ordered 2 times)
Last order date: 2026-02-18

```

על מנת לבדוק את נכונות הנתונים שחישה הפרוצדורה כתבנו שאילתה:

SQL Output Statistics

```

SELECT
    os.Supplier_Id,
    os.Order_Id,
    o.Order_Date,
    o.Delivery_Date,
    op.Product_Id
FROM
    Order_Supplier os
JOIN
    Orders o ON os.Order_Id = o.Order_Id
LEFT JOIN
    Order_Product op ON o.Order_Id = op.Order_Id
WHERE
    os.Supplier_Id = 101
    AND o.Order_Id IN (201, 601);

```

	SUPPLIER_ID	ORDER_ID	ORDER_DATE	DELLIVERY_DATE	PRODUCT_ID
1	101	201	18/02/2026	08/11/2030	101
2	101	601	08/10/2024	04/08/2030	101
3	101	201	18/02/2026	08/11/2030	201
4	101	601	08/10/2024	04/08/2030	201
5	101	201	18/02/2026	08/11/2030	301
6	101	201	18/02/2026	08/11/2030	501
7	101	201	18/02/2026	08/11/2030	601
8	101	601	08/10/2024	04/08/2030	601
9	101	201	18/02/2026	08/11/2030	701
10	101	601	08/10/2024	04/08/2030	701
11	101	601	08/10/2024	04/08/2030	801

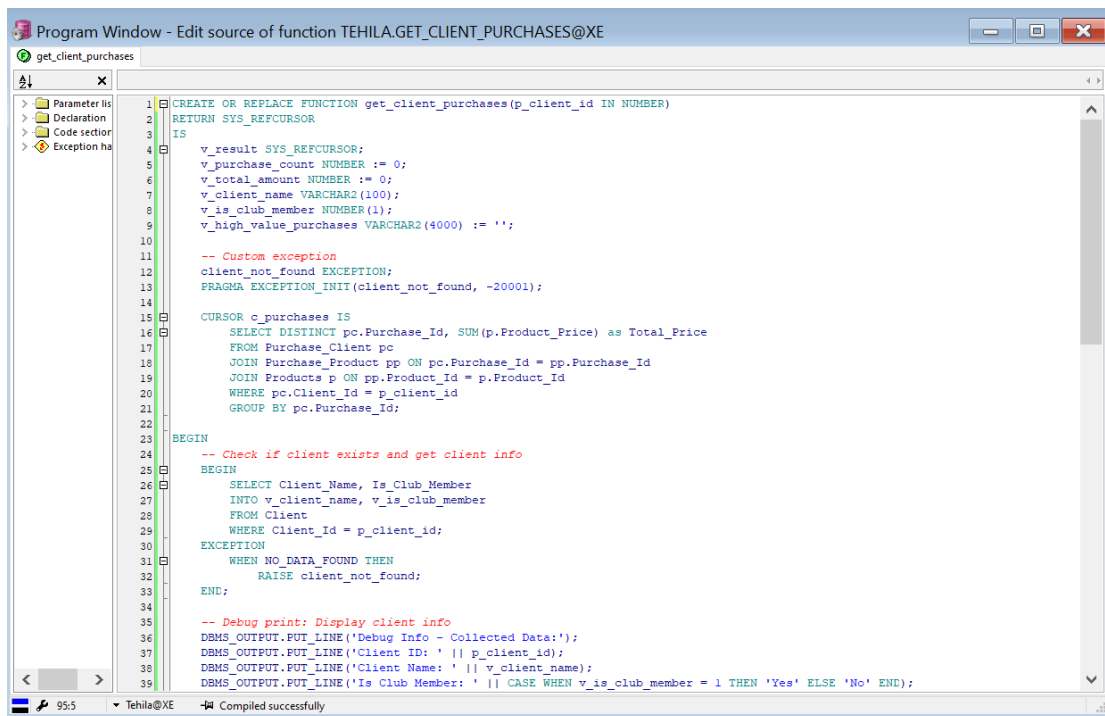
אכן רואים שהפרטים נכונים.

פונקציה #1

פונקציה המחשבת עבור מספר לקוח נתון את סך כל הרכישות שביצע אי פעם.

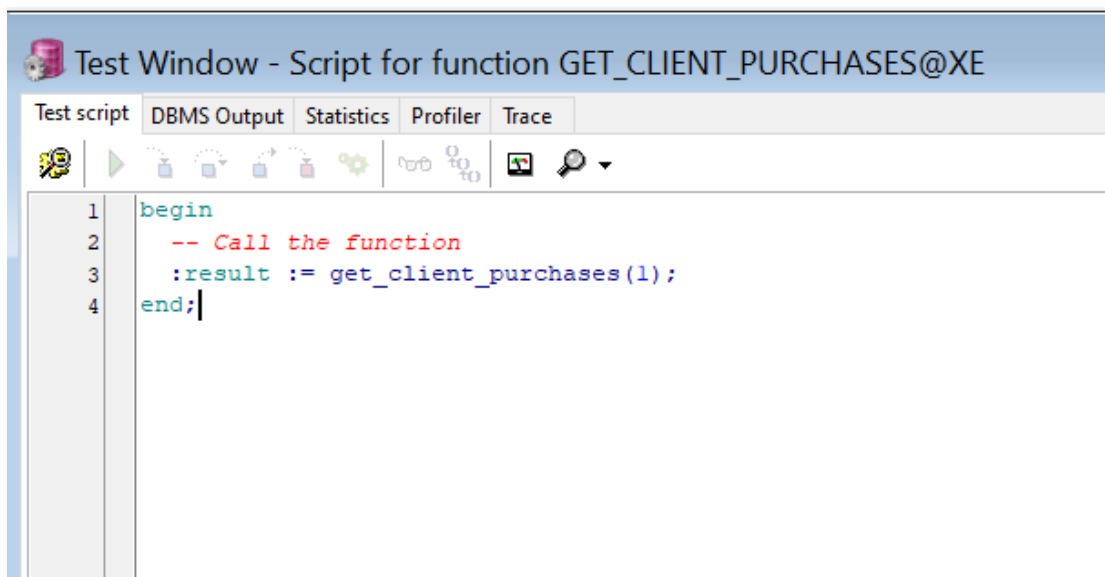
הפונקציה עושה שימוש בערך מוחזר כ ref cursor, לולאות, תנאים, פקודות DML, וכמובן EXCEPTION בעת הצורך.

(הקוד המלא כמובן מצורף כקובץ נפרד)



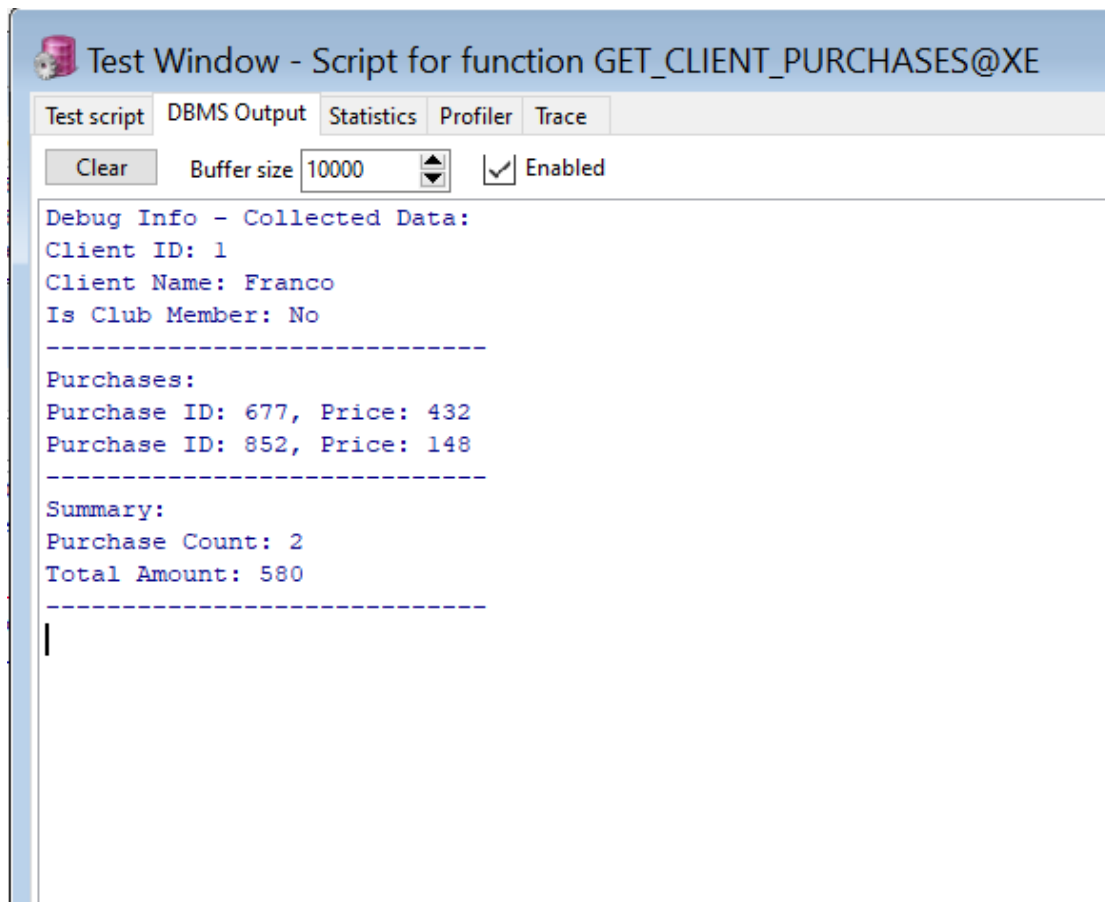
```
1 CREATE OR REPLACE FUNCTION get_client_purchases(p_client_id IN NUMBER)
2 RETURN SYS_REFCURSOR
3 IS
4     v_result SYS_REFCURSOR;
5     v_purchase_count NUMBER := 0;
6     v_total_amount NUMBER := 0;
7     v_client_name VARCHAR2(100);
8     v_is_club_member NUMBER(1);
9     v_high_value_purchases VARCHAR2(4000) := '';
10
11     -- Custom exception
12     client_not_found EXCEPTION;
13     PRAGMA EXCEPTION_INIT(client_not_found, -20001);
14
15     CURSOR c_purchases IS
16         SELECT DISTINCT pc.Purchase_Id, SUM(p.Product_Price) as Total_Price
17         FROM Purchase_Client pc
18         JOIN Purchase_Product pp ON pc.Purchase_Id = pp.Purchase_Id
19         JOIN Products p ON pp.Product_Id = p.Product_Id
20         WHERE pc.Client_Id = p_client_id
21         GROUP BY pc.Purchase_Id;
22
23 BEGIN
24     -- Check if client exists and get client info
25     BEGIN
26         SELECT Client_Name, Is_Club_Member
27         INTO v_client_name, v_is_club_member
28         FROM Client
29         WHERE Client_Id = p_client_id;
30     EXCEPTION
31         WHEN NO_DATA_FOUND THEN
32             RAISE client_not_found;
33     END;
34
35     -- Debug print: Display client info
36     DBMS_OUTPUT.PUT_LINE('Debug Info - Collected Data:');
37     DBMS_OUTPUT.PUT_LINE('Client ID: ' || p_client_id);
38     DBMS_OUTPUT.PUT_LINE('Client Name: ' || v_client_name);
39     DBMS_OUTPUT.PUT_LINE('Is Club Member: ' || CASE WHEN v_is_club_member = 1 THEN 'Yes' ELSE 'No' END);
```

הרצת הפונקציה:



```
1 begin
2     -- Call the function
3     :result := get_client_purchases(1);
4 end;
```

כיוון שהערך המוחזר הינו מסוג REF CURSOR ביצענו הדפסה לפני היציאה מהפונקציה על מנת להראות שאכן הפונקציה עובדת ומחשבת כראוי את הנתונים:



The screenshot shows the 'Test Window' for the function 'GET_CLIENT_PURCHASES@XE'. The window has tabs for 'Test script', 'DBMS Output', 'Statistics', 'Profiler', and 'Trace'. The 'DBMS Output' tab is selected, showing the following output:

```
Debug Info - Collected Data:
Client ID: 1
Client Name: Franco
Is Club Member: No
-----
Purchases:
Purchase ID: 677, Price: 432
Purchase ID: 852, Price: 148
-----
Summary:
Purchase Count: 2
Total Amount: 580
-----
```

כתבנו שאילתה שתציג לנו את כל הרכישות עבור לקוח על מנת לראות שאכן הנתונים שהפונקציה מציגה נכונים:

SQL Window - SELECT pc.Client_Id, c.Client_Name, p.Purchase_Id, pp.Product_Id, pr.Product_Name, pr.Product_Pr ...

SQL Output Statistics

```

SELECT
  pc.Client_Id,
  c.Client_Name,
  p.Purchase_Id,
  pp.Product_Id,
  pr.Product_Name,
  pr.Product_Price,
  p.Purchase_Date
FROM
  Purchase p
JOIN Purchase_Client pc ON p.Purchase_Id = pc.Purchase_Id
JOIN Client c ON pc.Client_Id = c.Client_Id
JOIN Purchase_Product pp ON p.Purchase_Id = pp.Purchase_Id
JOIN Products pr ON pp.Product_Id = pr.Product_Id

WHERE pc.Client_Id = 1
ORDER BY
  pc.Client_Id, p.Purchase_Id, pp.Product_Id;

```

	CLIENT_ID	CLIENT_NAME	PURCHASE_ID	PRODUCT_ID	PRODUCT NAME	PRODUCT PRICE	PURCHASE DATE
1	1	Franco	677	230	Best Buy Co.	30	21/08/2024
2	1	Franco	677	312	Progressive Des	12	21/08/2024
3	1	Franco	677	384	Blue Ocean Soft	84	21/08/2024
4	1	Franco	677	588	Unica	88	21/08/2024
5	1	Franco	677	677	Creditors Inter	77	21/08/2024
6	1	Franco	677	736	Unit	36	21/08/2024
7	1	Franco	677	819	Balchem	19	21/08/2024
8	1	Franco	677	886	Integrated Deci	86	21/08/2024
9	1	Franco	852	229	CLT Meetings In	29	22/05/2025
10	1	Franco	852	628	Peerless Manufa	28	22/05/2025
11	1	Franco	852	739	Infinity Softwa	39	22/05/2025
12	1	Franco	852	852	Market First	52	22/05/2025

ניתן לראות שאכן החישוב מדויק ונכון.

במידה ונזין מספר לקוח לא נכון תיזרק חריגה:

```

-- Custom exception
client_not_found EXCEPTION;
PRAGMA EXCEPTION_INIT(client_not_found, -20001);

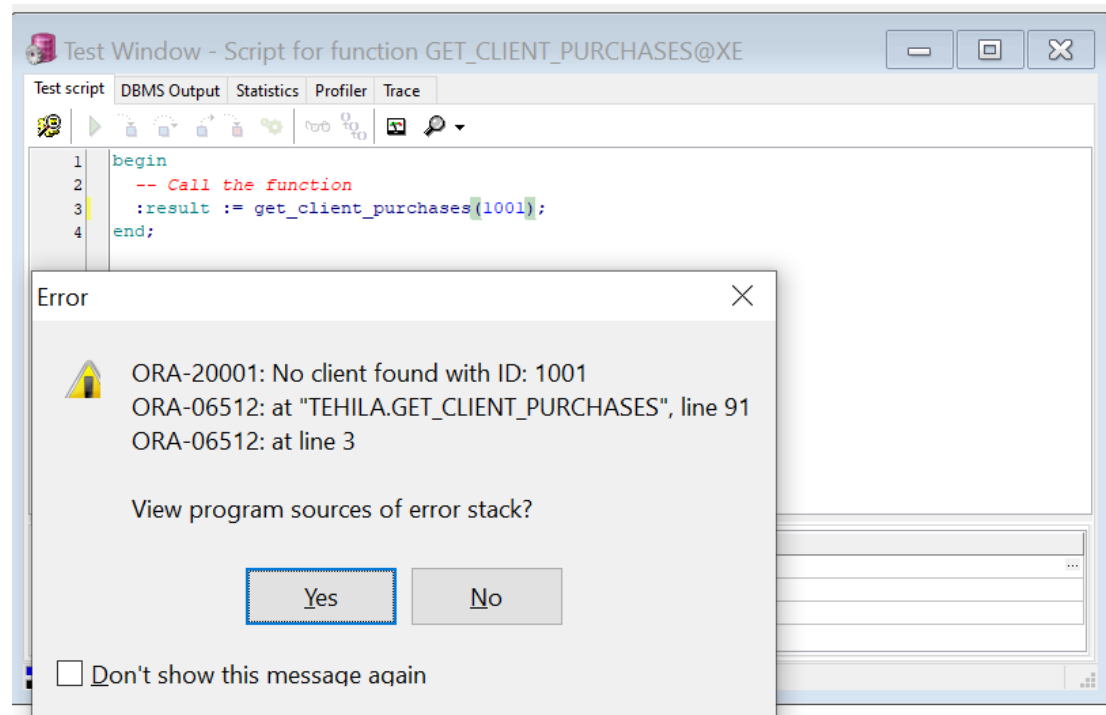
```

```

EXCEPTION
  WHEN client_not_found THEN
    DBMS_OUTPUT.PUT_LINE('Error: No client found with ID: ' || p_client_id);
    RAISE_APPLICATION_ERROR(-20001, 'No client found with ID: ' || p_client_id);
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('Error: ' || SQLERRM);
    RAISE_APPLICATION_ERROR(-20002, 'An error occurred: ' || SQLERRM);
END;

```

דוגמה להרצה וקבלת שגיאה:



פונקציה #2

הפונקציה מיועדת לאחזר את המוצרים המובילים (הנמכרים ביותר) בקטגוריה מסוימת.

פרמטרים מתקבלים:

1. p_category_id: מספר המזהה של הקטגוריה.

2. p_limit: מספר המוצרים המובילים לאחזר.

ערך מוחזר: הפונקציה מחזירה SYS_REFCURSOR המכיל את רשימת המוצרים המובילים.

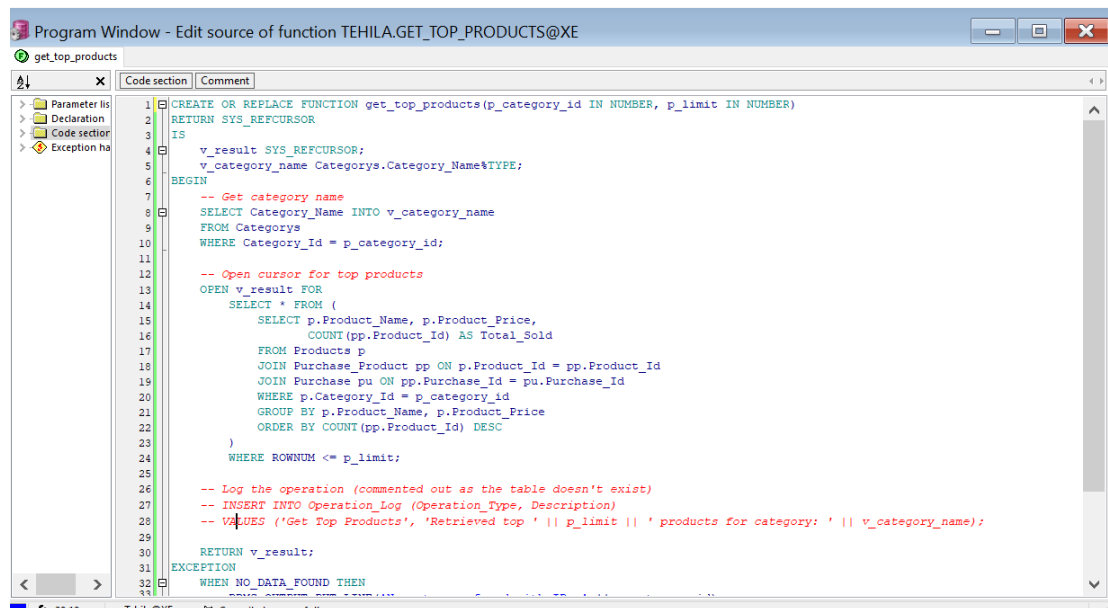
הפונקציה מאחזרת את שם הקטגוריה לפי המזהה שהתקבל. היא מבצעת שאילתה מורכבת לאיתור המוצרים המובילים בקטגוריה, מסודרים לפי כמות המכירות, מגבילה את התוצאות למספר שהתקבל כפרמטר ומחזירה את התוצאות ב-CURSOR.

הפונקציה מחזירה SYS_REFCURSOR, אמנם אין שימוש בלולאות מפורשות, אך השאילתה מבצעת איטרציה על הנתונים.

יש טיפול בשגיאות עבור NO_DATA_FOUND ו-OTHERS.

הפונקציה משתמשת בטיפוסי נתונים של טבלאות (לדוגמה, Categories.Category_Name%TYPE).

הפונקציה: (הקובץ המלא מצורף בנפרד כמובן)



```
1 CREATE OR REPLACE FUNCTION get_top_products(p_category_id IN NUMBER, p_limit IN NUMBER)
2 RETURN SYS_REFCURSOR
3 IS
4     v_result SYS_REFCURSOR;
5     v_category_name Categories.Category_Name%TYPE;
6 BEGIN
7     -- Get category name
8     SELECT Category_Name INTO v_category_name
9     FROM Categories
10    WHERE Category_Id = p_category_id;
11
12     -- Open cursor for top products
13     OPEN v_result FOR
14     (
15         SELECT p.Product_Name, p.Product_Price,
16                COUNT(pp.Product_Id) AS Total_Sold
17         FROM Products p
18         JOIN Purchase_Product pp ON p.Product_Id = pp.Product_Id
19         JOIN Purchase pu ON pp.Purchase_Id = pu.Purchase_Id
20         WHERE p.Category_Id = p_category_id
21         GROUP BY p.Product_Name, p.Product_Price
22         ORDER BY COUNT(pp.Product_Id) DESC
23     )
24    WHERE ROWNUM <= p_limit;
25
26     -- Log the operation (commented out as the table doesn't exist)
27     -- INSERT INTO Operation_Log (Operation_Type, Description)
28     -- VALUES ('Get Top Products', 'Retrieved top ' || p_limit || ' products for category: ' || v_category_name);
29
30     RETURN v_result;
31 EXCEPTION
32     WHEN NO_DATA_FOUND THEN
33         RAISE_APPLICATION_ERROR(-20000, 'Category not found');
34     WHEN OTHERS THEN
35         RAISE;
```


הרצת הפונקציה והדפסת הנתונים לאחר מכן:

```
de section | Statement
1 CREATE OR REPLACE FUNCTION get_top_products(p_category_id IN NUMBER, p_limit IN NUMBER)
2 RETURN SYS_REFCURSOR
3 IS
4     v_result SYS_REFCURSOR;
5     v_category_name Categories.Category_Name%TYPE;
6 BEGIN
7     -- Get category name
8     SELECT Category_Name INTO v_category_name
9     FROM Categories
10    WHERE Category_Id = p_category_id;
11
12    -- Open cursor for top products
13    OPEN v_result FOR
14        SELECT * FROM (
15            SELECT p.Product_Name, p.Product_Price,
16                   COUNT(pp.Product_Id) AS Total_Sold
17            FROM Products p
18            JOIN Purchase_Product pp ON p.Product_Id = pp.Product_Id
19            JOIN Purchase pu ON pp.Purchase_Id = pu.Purchase_Id
20            WHERE p.Category_Id = p_category_id
21            GROUP BY p.Product_Name, p.Product_Price
22            ORDER BY COUNT(pp.Product_Id) DESC
23        )
24    WHERE ROWNUM <= p_limit;
25
26    -- Log the operation (commented out as the table doesn't exist)
27    -- INSERT INTO Operation_Log (Operation_Type, Description)
28    -- VALUES ('Get Top Products', 'Retrieved top ' || p_limit || ' products for category: ' || v_category_name);
29
30    RETURN v_result;
31 EXCEPTION
32 WHEN NO_DATA_FOUND THEN
33     DBMS_OUTPUT.PUT_LINE('No category found with ID: ' || p_category_id);
34     RETURN NULL;
35 WHEN OTHERS THEN
36     DBMS_OUTPUT.PUT_LINE('An error occurred: ' || SQLERRM);
37     RETURN NULL;
38 END;
```

פלט לאחר הרצה :

```
Top 2 products in category 110:
Lydian Trust - Price: 10, Total Sold: 2
Extra Mile Tran - Price: 10, Total Sold: 2
```

בדיקת נכונות הנתונים:

	PRODUCT_ID	PRODUCT_NAME	QUANTITY	PRODUCT_PRICE	CATEGORY_ID
1	110	Larkin Enterpri	10	10	110
2	210	American Pan &	10	10	110
3	310	Extra Mile Tran	10	10	110
4	410	Lydian Trust	10	10	110
5	510	Allegiant Banco	10	10	110
6	610	LogistiCare	10	10	110

תכנית #1

הערה חשובה!

את התוכניות הייתה אפשרות לשמור בכל מיני אופנים, אנחנו בחרנו לשמור אותן כפרוצדורות.

התוכנית משלבת מידע מהפונקציה `get_client_purchases` והפרוצדורה `analyze_supplier_orders` עם ניתוח נוסף כדי לספק תמונה מקיפה של הקשר בין לקוח ספציפי לספק ספציפי, ולהציע המלצות עסקיות על בסיס הנתונים.

1. מנתחת רכישות של לקוח ספציפי

- קוראת לפונקציה `get_client_purchases` ומקבלת את התוצאות.
- מציגה מידע על הלקוח: שם, מספר רכישות, סכום כולל, וחברות במועדון.
- מחשבת ומציגה את הסכום הממוצע לרכישה.

2. מנתחת הזמנות של ספק ספציפי

- קוראת לפרוצדורה `analyze_supplier_orders`.

3. מבצעת ניתוח נוסף:

- סופרת כמה מוצרים מהספק הספציפי נרכשו על ידי הלקוח.

4. מציגה תובנות מעניינות:

○ מציעה חברות במועדון ללקוח אם יש לו היקף רכישות גבוה ואינו חבר.

○ מציעה להגדיל הזמנות מהספק אם הלקוח קנה הרבה מהמוצרים שלו.

5. מטפלת בשגיאות אפשריות ומציגה הודעת שגיאה במקרה הצורך.

התכנית: (הקובץ המלא מצורף בנפרד)

```
1 DECLARE
2     -- Variables for get_client_purchases
3     v_client_id NUMBER := 1; -- Assume we have a client with ID 1
4     v_client_cursor SYS_REFCURSOR;
5     v_client_name VARCHAR2(100);
6     v_purchase_count NUMBER;
7     v_total_amount NUMBER;
8     v_is_club_member VARCHAR2(3);
9
10    -- Variables for analyze_supplier_orders
11    v_supplier_id NUMBER := 130; -- Assume we have a supplier with ID 1
12
13    -- Additional variables for interesting analysis
14    v_avg_purchase_amount NUMBER;
15    v_supplier_product_count NUMBER;
16 BEGIN
17     DBMS_OUTPUT.PUT_LINE('=== Client Purchase Analysis ===');
18
19     -- Call get_client_purchases function
20     v_client_cursor := get_client_purchases(v_client_id);
21
22     -- Fetch results from the cursor
23     FETCH v_client_cursor INTO v_client_name, v_purchase_count, v_total_amount, v_is_club_member;
24     CLOSE v_client_cursor;
25
26     -- Display results
27     DBMS_OUTPUT.PUT_LINE('Client Name: ' || v_client_name);
28     DBMS_OUTPUT.PUT_LINE('Total Purchases: ' || v_purchase_count);
29     DBMS_OUTPUT.PUT_LINE('Total Amount Spent: $' || TO_CHAR(v_total_amount, '999,999.99'));
30     DBMS_OUTPUT.PUT_LINE('Club Member: ' || v_is_club_member);
31
32     -- Calculate average purchase amount
33     IF v_purchase_count > 0 THEN
34         v_avg_purchase_amount := v_total_amount / v_purchase_count;
35         DBMS_OUTPUT.PUT_LINE('Average Purchase Amount: $' || TO_CHAR(v_avg_purchase_amount, '999,999.99'));
36     END IF;
37
38     DBMS_OUTPUT.PUT_LINE('');
```

לאחר הרצה: (הגדרנו שמספר רכישות שיוגדר גבוה יהיה 1 ומעלה כדי לראות את ההדפסה, זהו תנאי שניתן לשנות בכל עת כמובן)

```
=== Client Purchase Analysis ===
Debug Info - Collected Data:
Client ID: 1
Client Name: Franco
Is Club Member: No

-----
Purchases:
Purchase ID: 677, Price: 432
Purchase ID: 852, Price: 148
-----
Summary:
Purchase Count: 2
Total Amount: 580
-----
Client Name: Franco
Total Purchases: 2
Total Amount Spent: $      580.00
Club Member: No
Average Purchase Amount: $      290.00

=== Supplier Order Analysis ===
Analyzing orders for supplier: Christian
Analysis for supplier: Christian
Total orders: 1
Total products ordered: 5
Most ordered product: Procter & Gambl (Ordered 1 times)
Last order date: 2024-05-05

=== Interesting Insights ===
Number of products from Supplier 130 purchased by Client 1: 1
Suggestion: Consider offering Club Membership to Franco based on their high purchase volume.
```

תכנית #2

זוהי תוכנית המתמקדת בעדכון מחירים וניתוח מכירות בקטגוריה ספציפית, תוך שימוש בפרוצדורות ופונקציות חיצוניות לביצוע העיבוד העיקרי.

לאחר שהתכנית כמובן מגדירה את המשתנים הרלוונטיים היא:

1. מעדכנת מחירים בקטגוריה :

◦ קוראת לפרוצדורה `update_category_prices` עדכון המחירים.

2. מנתחת מוצרים מובילים :

◦ קוראת לפונקציה `get_top_products` לקבלת 5 (או מספר נבחר אחר) המוצרים המובילים בקטגוריה.

3. מציגה תוצאות :

◦ מציגה בלולאה את המוצרים המובילים עם פרטיהם (שם, מחיר חדש, כמות שנמכרה).

4. מטפלת בשגיאות :

◦ כוללת טיפול בשגיאות בסיסי ומציגה הודעת שגיאה במקרה הצורך.

התכנית: (הקובץ המלא מצורף בנפרד)

```
1 DECLARE
2   v_category_id NUMBER := 102; -- Assume this is the ID of some category
3   v_supplier_id NUMBER := 101; -- Assume this is the ID of some supplier
4   v_price_increase NUMBER := 15; -- Price increase
5   v_top_products SYS_REFCURSOR;
6   v_product_name Products.Product_Name%TYPE;
7   v_product_price Products.Product_Price%TYPE;
8   v_total_sold NUMBER;
9 BEGIN
10  DBMS_OUTPUT.PUT_LINE('Starting analysis and processing for category ' || v_category_id || ' and supplier ' || v_supplier_id);
11
12  -- Call procedure to update prices in the category
13  update_category_prices(v_category_id, v_price_increase);
14
15  -- Get top products in the category after the update
16  v_top_products := get_top_products(v_category_id, 5);
17
18  DBMS_OUTPUT.PUT_LINE('Top products in the category after price update:');
19  LOOP
20    FETCH v_top_products INTO v_product_name, v_product_price, v_total_sold;
21    EXIT WHEN v_top_products%NOTFOUND;
22    DBMS_OUTPUT.PUT_LINE(v_product_name || ' - Price: ' || v_product_price || ', Amount sold: ' || v_total_sold);
23  END LOOP;
24  CLOSE v_top_products;
25
26  DBMS_OUTPUT.PUT_LINE('Completed analysis and processing for category');
27 EXCEPTION
28  WHEN OTHERS THEN
29    DBMS_OUTPUT.PUT_LINE('An error occurred: ' || SQLERRM);
30 END;
```

לאחר ההרצה:

```
Starting analysis and processing for category 102 and supplier 101
Price increase percentage: 15
Product: Quality Assured
Old price: 10
Multiplier: 1.15
New price before rounding: 11.50
New price after rounding: 11.5
Updated product: Quality Assured, Old price: 10, New price: 11.5
-----
Product: Unica
Old price: 2
Multiplier: 1.15
New price before rounding: 2.30
New price after rounding: 2.3
Updated product: Unica, Old price: 2, New price: 2.3
-----
Product: Maverick Techno
Old price: 14
Multiplier: 1.15
New price before rounding: 16.10
New price after rounding: 16.1
Updated product: Maverick Techno, Old price: 14, New price: 16.1
-----
Product: SM Consulting
Old price: 2
Multiplier: 1.15
New price before rounding: 2.30
New price after rounding: 2.3
Updated product: SM Consulting, Old price: 2, New price: 2.3
-----
```

```
Updated product: Maverick Techno, Old price: 14, New price: 16.1
-----
Product: SM Consulting
Old price: 2
Multiplier: 1.15
New price before rounding: 2.30
New price after rounding: 2.3
Updated product: SM Consulting, Old price: 2, New price: 2.3
-----
Product: Limited Brands
Old price: 3
Multiplier: 1.15
New price before rounding: 3.45
New price after rounding: 3.45
Updated product: Limited Brands, Old price: 3, New price: 3.45
-----
Product: Strategic Produ
Old price: 35
Multiplier: 1.15
New price before rounding: 40.25
New price after rounding: 40.25
Updated product: Strategic Produ, Old price: 35, New price: 40.25
-----
Total updated products in category officiis. : 6
Top products in the category after price update:
Strategic Produ - Price: 40, Amount sold: 1
Maverick Techno - Price: 16, Amount sold: 1
Completed analysis and processing for category
|
```

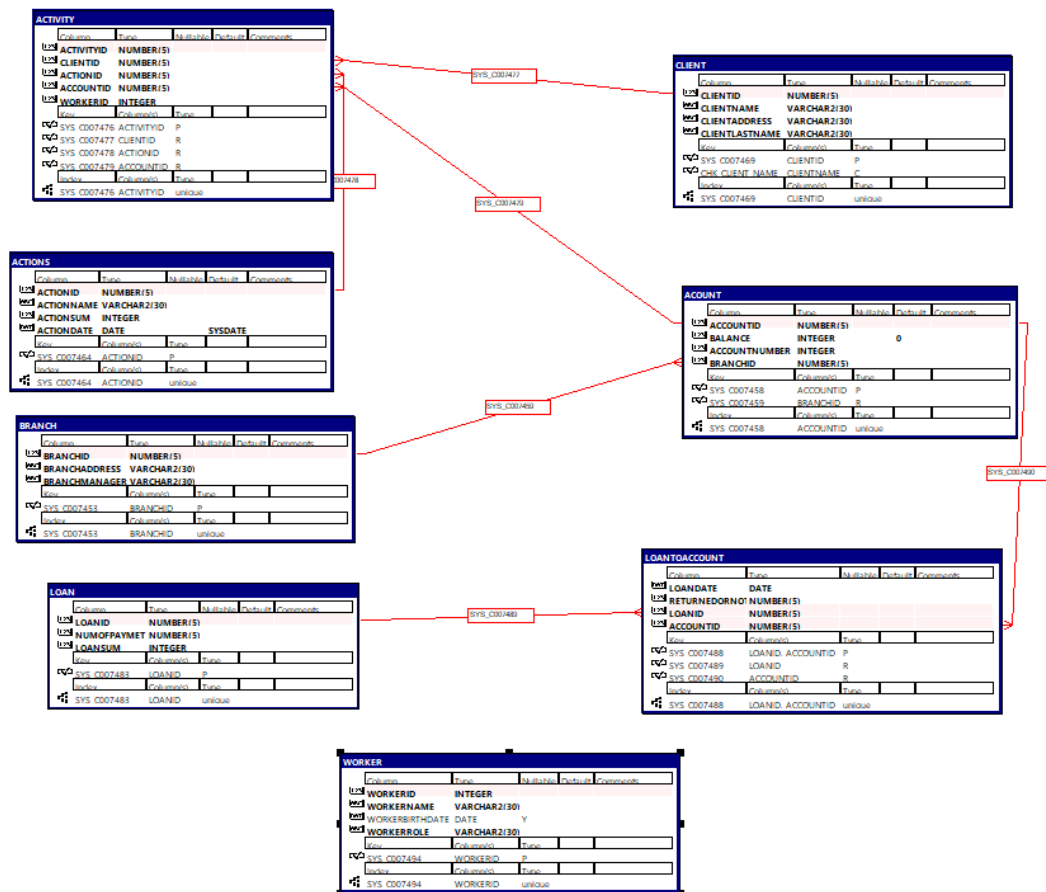
שלב 4

שחזרנו את הקובץ גיבוי של הפרויקט , לאחר מכן פתחנו בטבלאות DSD ולפי הנתונים בנינו את דיאגרמת ERD.

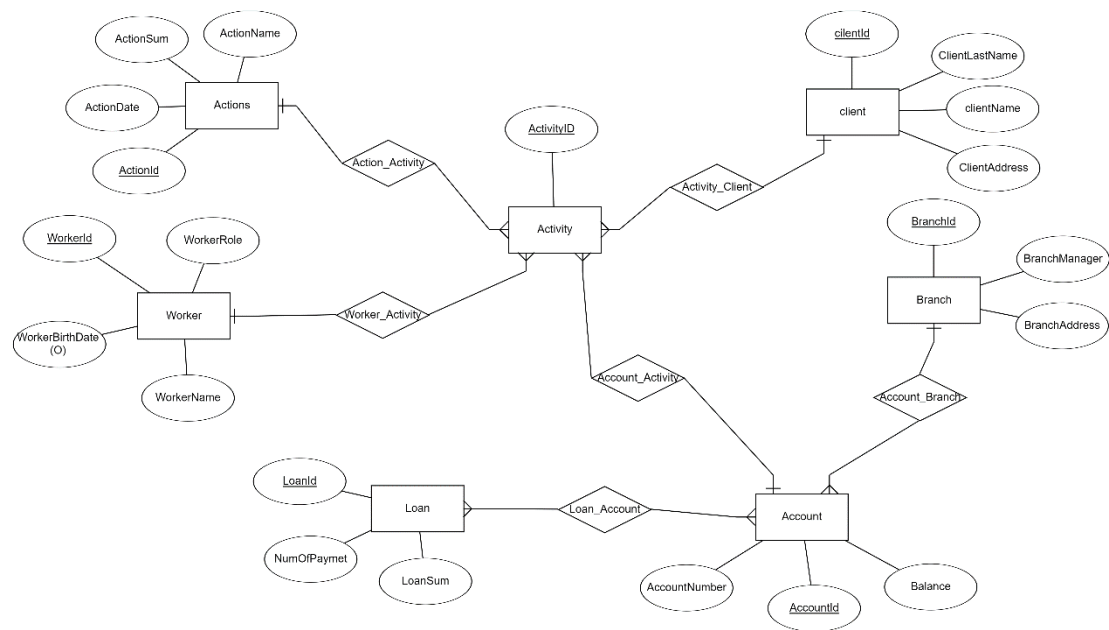
שינינו את השם של הטבלאות שלנו מCLIENT ל client_s

ומ worker_s ל worker_s על מנת שנוכל לייצר את הטבלאות של הלקוח והעובד של הבנק.

:DSD DIAGRAM

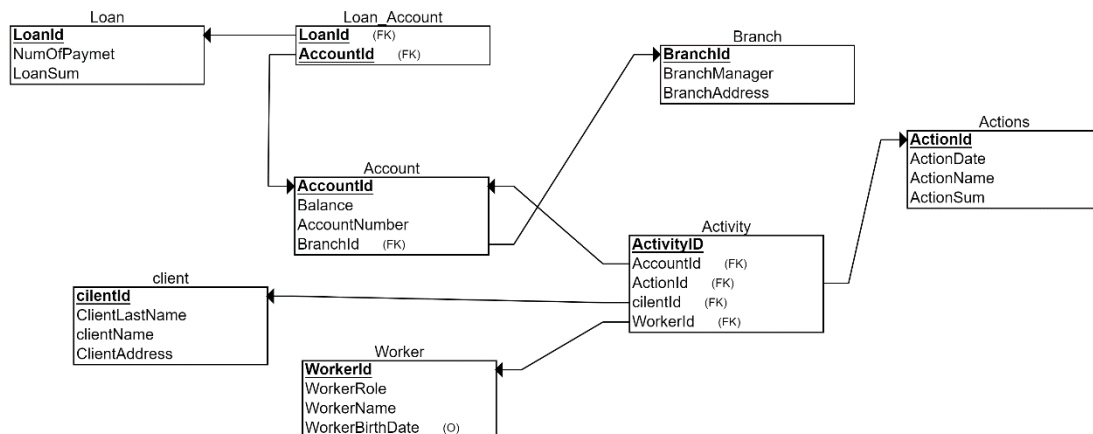


:ERD DIAGRAM



DSD DIAGRAM בתצוגה נוספת

אחרי יצירת ERD DIAGRAM:



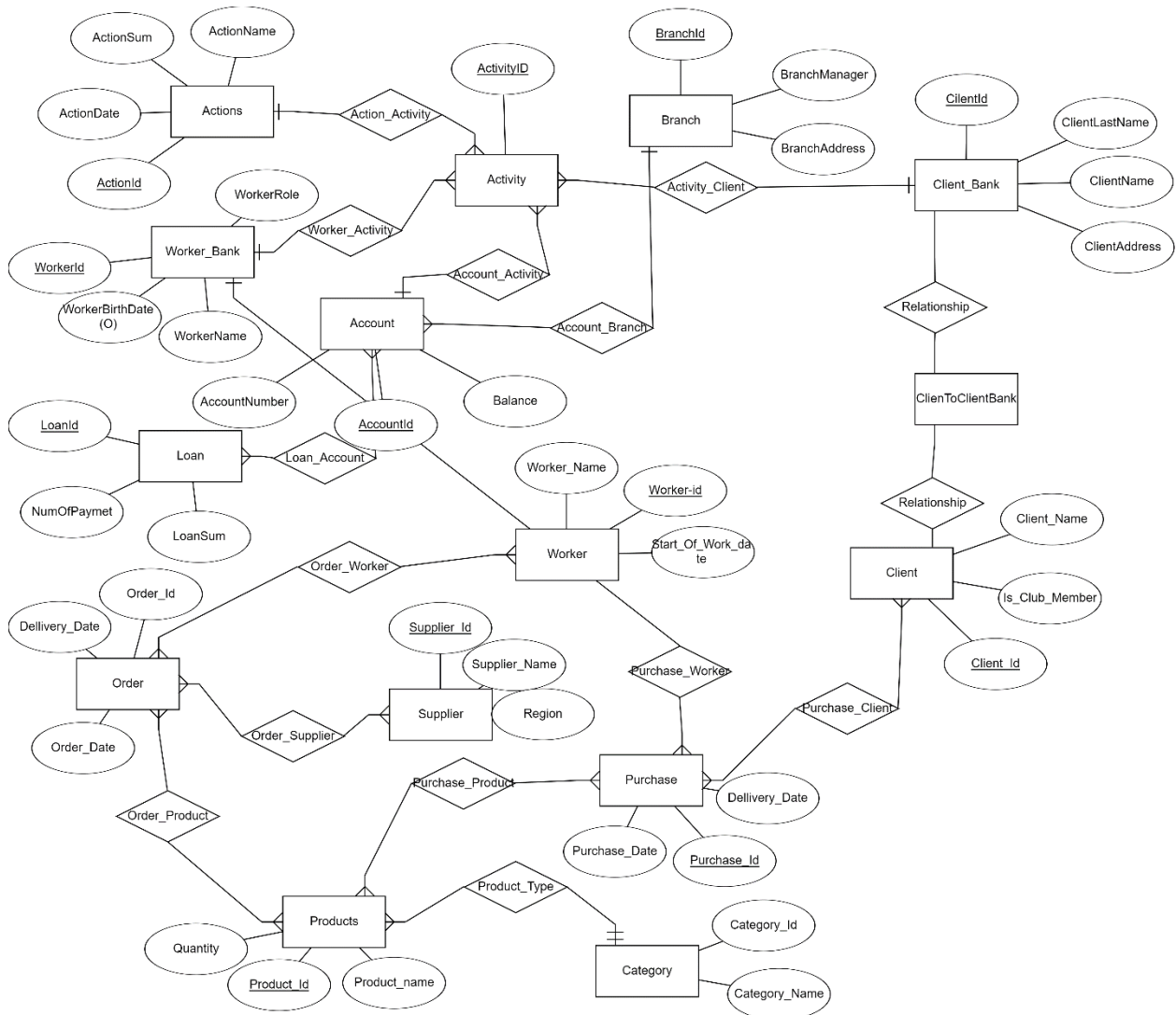
יצרנו את טבלה נוספת שמאחדת בין שני הפרויקטים.

כפי שניתן לראות איחדנו את הלקוח של הבנק ללקוח של החנות:

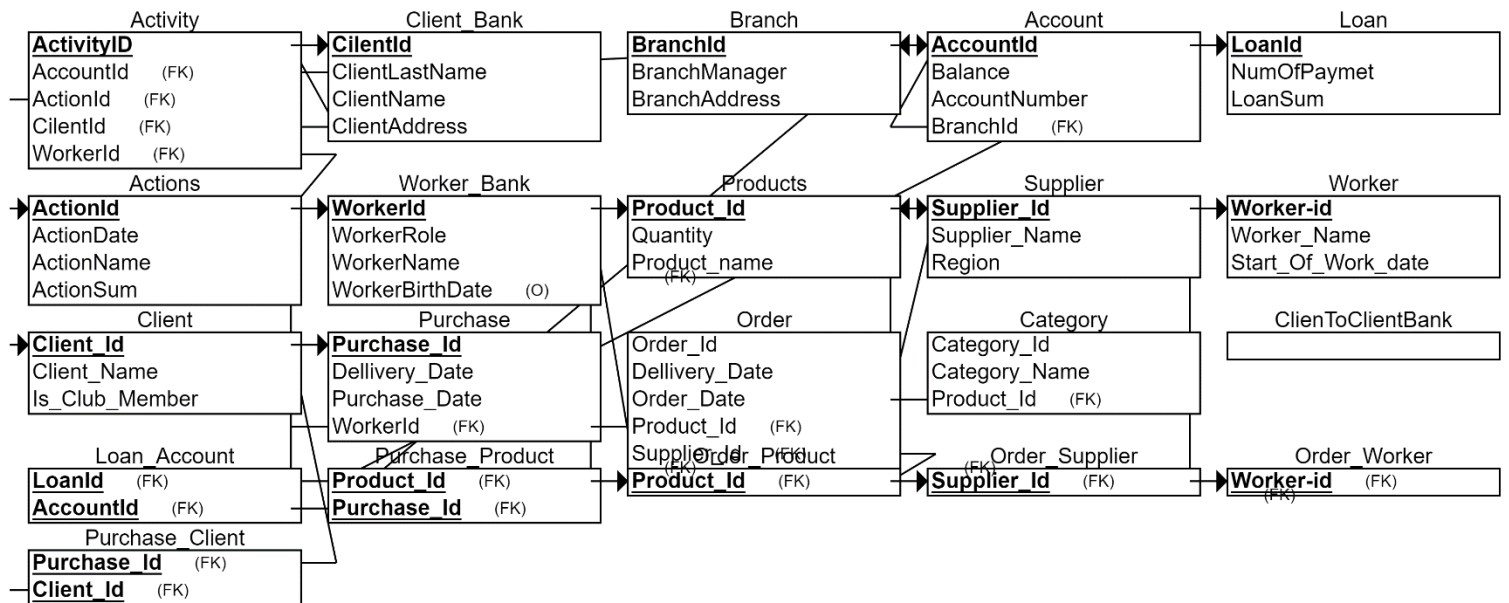
```

CREATE TABLE ClientToClientBank
(
    CLIENT_ID NUMERIC(5) NOT NULL,
    ClientID NUMERIC(5) NOT NULL,
    PRIMARY KEY (CLIENT_ID),
    FOREIGN KEY (ClientID) REFERENCES Client(ClientID),
    FOREIGN KEY (CLIENT_ID) REFERENCES Client_s(CLIENT_ID)
);
    
```

טבלת ERD משותף:



טבלת DSD משותף:



VIEWS

המבט ראשון יצרנו על הפרויקט שלנו :

המבט הזה, ClientPurchaseWorkerOrder, יוצר טבלה וירטואלית המשלבת מידע על לקוחות ורכישות עם מידע על עובדים והזמנות:

1. החלק הראשון מחבר מידע על לקוחות ורכישותיהם.

2. החלק השני (אחרי UNION ALL) מוסיף מידע על עובדים וההזמנות שהם טיפלו בהן.

התוצאה היא טבלה אחת המציגה:

- פרטי לקוחות ורכישותיהם (כולל תאריכי רכישה ומשלוח)

- פרטי עובדים וההזמנות שטיפלו בהן (כולל תאריכי הזמנה)

המבט מאפשר לראות את כל המידע הזה יחד, אך שדות שאינם רלוונטיים לכל חלק מוגדרים כ-NULL.

```
CREATE VIEW ClientPurchaseWorkerOrder AS
SELECT
    c.Client_Id,
    c.Client_Name,
    c.Is_Club_Member,
    p.Purchase_Id,
    p.Purchase_Date,
    p.Dellivery_Date,
    NULL AS Worker_Id,
    NULL AS Worker_Name,
    NULL AS Order_Id,
    NULL AS Order_Date
FROM
    Client_s c
JOIN
    Purchase_Client pc ON c.Client_Id = pc.Client_Id
JOIN
    Purchase p ON pc.Purchase_Id = p.Purchase_Id

UNION ALL

SELECT
    NULL AS Client_Id,
    NULL AS Client_Name,
    NULL AS Is_Club_Member,
    NULL AS Purchase_Id,
    NULL AS Purchase_Date,
    NULL AS Dellivery_Date,
    w.Worker_Id,
    w.Worker_Name,
    o.Order_Id,
    o.Order_Date
FROM
    Worker_s w
JOIN
    Order_Worker ow ON w.Worker_Id = ow.Worker_Id
JOIN
    Orders o ON ow.Order_Id = o.Order_Id;
```

בדיקת תקינות של המבט, כתבנו שתי שאילתות שבהם עשינו קריאה למבט :

```

SELECT
  Client_Id,
  Client_Name,
  Is_Club_Member,
  Purchase_Id,
  Purchase_Date,
  Delivery_Date
FROM
  ClientPurchaseWorkerOrder
WHERE
  Client_Id IS NOT NULL;

```

	CLIENT_ID	CLIENT_NAME	IS_CLUB_MEMBER	PURCHASE_ID	PURCHASE_DATE	DELIVERY_DATE
1	1	Franco	0	677	21/08/2024	09/04/2024
2	1	Franco	0	852	22/05/2025	17/12/2026
3	2	Chazz	0	843	17/10/2029	14/03/2024
4	3	Rita	1	814	22/07/2029	05/08/2026
5	5	Gabrielle	1	319	08/02/2025	02/05/2030
6	9	Lila	0	208	09/12/2025	18/03/2025
7	9	Lila	0	605	08/06/2030	26/07/2029
8	9	Lila	0	611	18/06/2026	03/08/2026
9	13	Sinead	0	814	22/07/2029	05/08/2026
10	19	Sheryl	0	123	28/02/2030	06/10/2025
11	22	Tommy	0	183	27/02/2027	28/05/2024
12	23	Charles	0	611	18/06/2026	03/08/2026
13	24	Johnnie	0	388	23/12/2026	07/05/2029
14	24	Johnnie	0	811	11/04/2028	05/04/2030
15	26	Shelby	1	676	29/05/2025	20/01/2030
16	26	Shelby	1	814	22/07/2029	05/08/2026
17	27	Hikaru	1	395	04/01/2027	22/03/2027
18	28	Kenneth	1	145	11/01/2025	29/07/2029
19	28	Kenneth	1	911	14/01/2030	06/06/2025
20	29	Dermot	0	863	24/07/2025	14/03/2026
21	35	Gordie	1	377	04/10/2025	11/08/2024
22	36	Juliana	0	162	20/05/2024	26/03/2028
23	36	Juliana	0	676	29/05/2025	20/01/2030
24	37	Gavin	1	863	24/07/2025	14/03/2026
25	45	Cherry	0	348	28/03/2025	27/01/2025
26	48	Linda	1	208	09/12/2025	18/03/2025

```

SELECT
  Worker_Id,
  Worker_Name,
  Order_Id,
  Order_Date
FROM
  ClientPurchaseWorkerOrder
WHERE
  Worker_Id IS NOT NULL;

```

	WORKER_ID	WORKER_NAME	ORDER_ID	ORDER_DATE
1	657	pureLwKil	557	06/04/2028
2	640	fBQjCbHYDF	540	18/01/2029
3	131	VVZPsTNsdu	331	29/04/2025
4	381	WenaOnSMuX	181	16/08/2030
5	381	WenaOnSMuX	381	23/03/2027
6	172	t5fOPjzCuC	272	16/02/2024
7	172	t5fOPjzCuC	572	07/05/2027
8	289	LbSDWHroML	889	06/03/2027
9	541	XSMzHonvPn	641	25/07/2026
10	844	yBETyLSLE	444	24/07/2025
11	855	LXIZoEKAZT	555	20/03/2024
12	744	LEolmByKar	444	24/07/2025
13	361	ziVrVEBGsu	961	08/12/2030
14	361	ziVrVEBGsu	561	14/03/2025
15	361	ziVrVEBGsu	361	02/07/2027
16	361	ziVrVEBGsu	261	11/07/2030
17	209	qBDWviOnPd	309	26/02/2025
18	591	oTfsXBdyST	191	08/09/2030
19	591	oTfsXBdyST	991	10/04/2026
20	355	OcpEYvkFoC	555	20/03/2024
21	790	LeXgaFhLR	490	07/10/2030
22	739	YulQgWssZ	139	14/03/2029
23	944	RzphUKGOv	444	24/07/2025
24	518	jnguVFOEtX	418	03/05/2027
25	518	jnguVFOEtX	318	30/09/2029
26	188	igFPQwpXZJ	388	12/10/2024
27	618	FQsmAmfyvD	418	03/05/2027
28	618	FQsmAmfyvD	318	30/09/2029

Michal@XE 28 rows selected in 0.062 seconds (more...)

יצרנו מבט נוסף לפרויקט השני :

המבט הזה יוצר טבלה וירטואלית בשם ClientActivityworker שמשלבת מידע מארבע טבלאות:

1. החלק הראשון מחבר מידע על לקוחות, פעולות ועובדים שביצעו פעולות.

2. החלק השני (אחרי UNION ALL) מוסיף עובדים שלא ביצעו אף פעולה.

התוצאה היא טבלה מקיפה המציגה את כל הפעילויות של הלקוחות והעובדים, כולל עובדים ללא פעילות.

```
CREATE VIEW ClientActivityworker AS
SELECT
    c.ClientId,
    c.ClientName,
    c.ClientLastName,
    a.ActionId,
    a.ActionName,
    a.ActionSum,
    a.ActionDate,
    w.WorkerId,
    w.WorkerName,
    w.WorkerRole
FROM
    Client c
JOIN
    activity ac ON c.ClientId = ac.ClientId
JOIN
    Actions a ON ac.ActionId = a.ActionId
JOIN
    Worker w ON ac.WorkerId = w.WorkerId

UNION ALL

SELECT
    NULL AS ClientId,
    NULL AS ClientName,
    NULL AS ClientLastName,
    NULL AS ActionId,
    NULL AS ActionName,
    NULL AS ActionSum,
    NULL AS ActionDate,
    w.WorkerId,
    w.WorkerName,
    w.WorkerRole
FROM
    Worker w
WHERE
    w.WorkerId NOT IN (SELECT DISTINCT WorkerId FROM activity);
```

34:15 Tehila@XE Done in 0.016 seconds

בדקנו תקינות , עשינו שאילתה שמבצעת קריאה למבט :

```
select * from ClientActivityworker
```

	CLIENTID	CLIENTNAME	CLIENTLASTNAME	ACTIONID	ACTIONNAME	ACTIONSUM	ACTIONDATE	WORKERID	WORKERNAME	WORKERROLE
695	323	Maceo	Benet	268	Transferring Funds	200311	12/12/1986	339	Mint	Business Account Manager
696	81	Arnold	Delta	273	Transferring Funds	834124	13/04/2001	275	Eugene	Teller
697	384	Danny	Idle	273	Transferring Funds	834124	13/04/2001	352	Collective	Business Account Manager
698	406	Dianne	Spears	273	Transferring Funds	834124	13/04/2001	31	Kelli	Financial Advisor
699	261	Kyra	Remar	200	Transferring Funds	558726	06/09/1982	117	Gena	Customer Service Representativ
700	437	Denise	McCabe	43	Withdrawing Funds	68462	31/05/2010	114	Rosanna	Business Account Manager
701	353	Kyra	Pitney	273	Transferring Funds	834124	13/04/2001	243	Katrin	Teller
702	321	Jonny Lee	Van Shelton	200	Transferring Funds	558726	06/09/1982	169	Solomon	Credit Analyst
703	84	Brad	Madonna	43	Withdrawing Funds	68462	31/05/2010	363	Chuck	Credit Analyst
704	240	Vern	Unger	43	Withdrawing Funds	68462	31/05/2010	351	Cuba	Credit Analyst
705	401	Humberto	O'Sullivan	268	Transferring Funds	200311	12/12/1986	395	Dwight	Financial Advisor
706	163	Temuera	Assante	273	Transferring Funds	834124	13/04/2001	263	Harris	Business Account Manager
707	115	Jet	Jonze	43	Withdrawing Funds	68462	31/05/2010	193	Meg	Credit Analyst
708	13	Sarah	Hatfield	141	Transferring Funds	319361	13/10/1991	55	Nigel	Customer Service Representativ
709								2	Ron	Teller
710								15	Julie	Customer Service Representativ
711								26	Mira	Customer Service Representativ
712								40	Udo	Business Account Manager
713								43	Famke	Customer Service Representativ
714								52	Mekhi	Investment Advisor
715								56	Brothers	Financial Advisor
716								61	Elle	Financial Advisor