TUGAS AKHIR

PELATIHAN DATABASE DESIGN AND PROGRAMMING WITH SQL

PERANCANGAN DESAIN DAN IMPLEMENTASI DATABASE ONLINE SHOP

Disusun oleh :

Nama	: Friska Cahya Yumanda
Nomor Registrasi	: 149294172100-673
Asal Universitas	: UPN Veteran Jakarta
Program Studi	: S1 Ekonomi Pembangunan

PROGRAM DIGITAL TALENT SCHOLARSHIP

KEMENTERIAN KOMUNIKASI DAN INFORMASI

UNIVERSITAS ISLAM NEGERI SYARIF HIDAYATULLAH JAKARTA

2022

KATA PENGANTAR

Penulis panjatkan puji syukur atas kehadirat Allah Subhanahu wa ta'ala karena dengan rahmat dan karunia-Nya, penulis dapat menyelesaikan tugas akhir dari kegiatan pelatihan "*Database Design and Programming with SQL*" yang diselenggarakan oleh perguruan tinggi UIN Syarif Hidayatullah Jakarta selaku mitra pelaksana program *digital talent scholarship* yang diselenggarakan oleh Kementerian Komunikasi dan Informasi yang bekerja sama dengan Oracle.

Dalam kesempatan ini, penulis ingin mengucapkan terima kasih yang sebesar-besarnya kepada pihak-pihak yang telah membantu penulis baik berupa kesempatan, ilmu dan pengalaman selama kegiatan pelatihan berlangsung. Ucapan terima kasih penulis sampaikan kepada:

1. Ibu Viva Arifin selaku pengajar pelatihan "Database Design and Programming with SQL" dari Universitas Islam Negeri Syarif Hidayatullah Jakarta.
2. Ibu Titik Inayah selaku koordinator kegiatan pelatihan "Database Design and Programming with SQL" dari Universitas Islam Negeri Syarif Hidayatullah Jakarta.

Penulis juga ingin mengucapkan terima kasih kepada kedua orang tua yang telah memberikan izin dan dukungan. Tugas akhir ini dibuat sebagai bentuk pertanggungjawaban dan pemenuhan syarat dari kegiatan pelatihan *Digital Talent Scholarship* yang telah penulis laksanakan pada perguruan tinggi penyelenggara UIN Syarif Hidayatullah Jakarta. Dalam pembuatan tugas akhir ini penulis menyadari betul bahwa masih terdapat banyak kekurangan, maka dari itu kritik dan saran yang membangun sangat diharapkan penulis untuk penyusunan laporan yang lebih baik di masa mendatang. Penulis berharap tulisan ini dapat memberi manfaat untuk pembaca.

Jakarta, 19 Agustus 2022

Penulis

DAFTAR ISI

DAFTAR TABEL

A. QUERY DAN HASIL RUNNING

1. CREATING TABLE AND CONSTRAINT

1.1. CREATING CUSTOMER TABLE (with constraint)

```
CREATE TABLE customer
(customer_id VARCHAR2(20) CONSTRAINT g_loc_customer_id_nn NOT NULL,
c_name VARCHAR2(20) CONSTRAINT g_loc_c_name_nn NOT NULL,
c_city VARCHAR(20),
discount NUMBER(10,2),
c_email VARCHAR2(20) UNIQUE CONSTRAINT g_loc_c_email_nn NOT NULL,
CONSTRAINT customer_customer_id_pk PRIMARY KEY(customer_id));
```

Table 1. Customer Table Structure

Column Name	Data Type	Nullable	Default	Primary Key
CUSTOMER_ID	VARCHAR2(20)	No	-	1
C_NAME	VARCHAR2(20)	No	-	-
C_CITY	VARCHAR2(20)	Yes	-	-
DISCOUNT	NUMBER(10,2)	Yes	-	-
C_EMAIL	VARCHAR2(20)	No	-	-

1.2. CREATE PRODUCT TABLE (with constraint)

```
CREATE TABLE product
(product_id VARCHAR2(20) CONSTRAINT g_loc_product_id_nn NOT NULL,
p_name VARCHAR(20),
quantity NUMBER(10) CONSTRAINT g_loc_quantity_nn NOT NULL,
price NUMBER(10,2) CONSTRAINT g_loc_price_nn NOT NULL,
CONSTRAINT product_product_id_pk PRIMARY KEY(product_id));
```

Table 2. Product Table Structure

Column Name	Data Type	Nullable	Default	Primary Key
PRODUCT_ID	VARCHAR2(20)	No	-	1
P_NAME	VARCHAR2(20)	Yes	-	-
QUANTITY	NUMBER(10,0)	No	-	-
PRICE	NUMBER(10,2)	No	-	-

1.3. CREATE ORDERING TABLE (with constraint)

```
CREATE TABLE ordering
(ordno VARCHAR2(20) CONSTRAINT g_loc_order_no_nn NOT NULL,
month DATE,
quantity NUMBER(10) CONSTRAINT g_loc_quantity_order_nn NOT NULL,
CONSTRAINT ordering_orderno_pk PRIMARY KEY(ordno),
customer_id VARCHAR(20) CONSTRAINT ordering_customer_id_fk
REFERENCES customer(customer_id),
product_id VARCHAR(20) CONSTRAINT ordering_product_id_fk
REFERENCES product(product_id));
```

Table 3. Ordering Table Structure

Column Name	Data Type	Nullable	Default	Primary Key
ORDNO	VARCHAR2(20)	No	-	1
MONTH	DATE	Yes	-	-
QUANTITY	NUMBER(10,0)	No	-	-
CUSTOMER_ID	VARCHAR2(20)	Yes	-	-
PRODUCT_ID	VARCHAR2(20)	Yes	-	-

2. DATA MANIPULATION (INSERT, ALTER, UPDATE)

2.1. INSERT MULTIPLE ROWS INTO PRODUCT TABLE AT ONCE (with Union All)

```
INSERT INTO product(product_id, p_name, quantity, price)
WITH p AS(
SELECT 'p01', 'pants', 20, 30.00 FROM dual UNION ALL
SELECT 'p02', 't-shirt', 10, 15.50 FROM dual UNION ALL
SELECT 'p03', 'shoes', 12, 35.00 FROM dual UNION ALL
SELECT 'p04', 'jacket', 5, 50.00 FROM dual UNION ALL
SELECT 'p05', 'glasses', 25, 70.75 FROM dual UNION ALL
SELECT 'p06', 'brecelet', 50, 5.00 FROM dual UNION ALL
SELECT 'p07', 'mask', 120, 1.50 FROM dual UNION ALL
SELECT 'p08', 'bag', 12, 25.00 FROM dual UNION ALL
SELECT 'p09', 'headphone', 30, 60.00 FROM dual UNION ALL
SELECT 'p10', 'shirt', 50, 1.15 FROM dual UNION ALL
SELECT 'p11', 'hat', 20, 10.99 FROM dual UNION ALL
SELECT 'p12', 'shorts', 5, 20.25 FROM dual UNION ALL
SELECT 'p13', 'earrings', 60, 35.50 FROM dual UNION ALL
SELECT 'p14', 'earphone', 10, 30.00 FROM dual UNION ALL
SELECT 'p15', 'sunglass', 20, 1.50 FROM dual)
SELECT * FROM p;
```

SELECT * FROM product

Table 4. All Data in Product Table

PRODUCT_ID	P_NAME	QUANTITY	PRICE
p01	pants	20	30
p02	t-shirt	10	15.5
p03	shoes	12	35
p04	jacket	5	50
p05	glasses	25	70.75
p06	brecelet	50	5
p07	mask	120	1.5
p08	bag	12	25
p09	headphone	30	60
p10	shirt	50	10.15
p11	hat	20	10.99
p12	shorts	5	20.25
p13	earrings	60	35.5
p14	earphone	10	30
p15	sunglass	20	40

2.2. UPDATE DATA IN PRODUCT TABLE

```
--update price value of shirt
UPDATE product
SET price = 10.15, p_name ='shirt'
WHERE product_id = 'p10';

--update price value of shirt
UPDATE product
SET price = 40, p_name ='sunglass'
WHERE product_id = 'p15';
```

SELECT product_id, p_name, price FROM product WHERE price = 10.15 OR price = 40;

Table 5. Two Rows in Product Table after Updated

PRODUCT_ID	P_NAME	PRICE
p10	shirt	10.15
p15	sunglass	40

2.3. INSERT MULTIPLE ROWS INTO CUSTOMER TABLE AT ONCE (with Union All)

```
INSERT INTO customer (customer_id, c_name, c_city, discount, c_email)
WITH c AS(
SELECT 'c001', 'Anisa', 'Jakarta Pusat', NULL, 'anisa@gmail.com' FROM dual UNION ALL
SELECT 'c002', 'Bagus', 'Jakarta Selatan', NULL, 'bagus@gmail.com' FROM dual UNION ALL
SELECT 'c003', 'Cita', 'Depok', 0.75, 'cita@gmail.com' FROM dual UNION ALL
SELECT 'c004', 'Dika', 'Tangerang', NULL, 'dika@gmail.com' FROM dual UNION ALL
SELECT 'c005', 'Fifi', 'Depok', NULL, 'fifi@gmail.com' FROM dual UNION ALL
SELECT 'c006', 'Lili', 'Jakarta Selatan', 0.10, 'lili@gmail.com' FROM dual UNION ALL
SELECT 'c007', 'Melina', 'Jakarta Timur', 0.40, 'melina@gmail.com' FROM dual UNION ALL
SELECT 'c008', 'Nanda', 'Jakarta Selatan', 0.25, 'nanda@gmail.com' FROM dual UNION ALL
SELECT 'c009', 'Gina', 'Tangerang', 1.00, 'gina@gmail.com' FROM dual UNION ALL
SELECT 'c010', 'Helmi', 'Jakarta Pusat', 0.50, 'helmi@gmail.com' FROM dual UNION ALL
SELECT 'c011', 'Iriana', 'Jakarta Selatan', 0.25, 'iriana@gmail.com' FROM dual UNION ALL
SELECT 'c012', 'Jeremy', 'Jakarta Barat', NULL , 'jeremy@gmail.com' FROM dual UNION ALL
SELECT 'c013', 'Kiara', 'Jakarta Selatan', NULL, 'kiara@gmail.com' FROM dual UNION ALL
SELECT 'c014', 'Okta', 'Bekasi', 1.50, 'okta@gmail.com' FROM dual)
SELECT * FROM c;
```

SELECT * FROM customer

Table 6. All Data in Customer Table

CUSTOMER_ID	C_NAME	C_CITY	DISCOUNT	C_EMAIL
c001	Anisa	Jakarta Pusat	-	anisa@gmail.com
c002	Bagus	Jakarta Selatan	-	bagus@gmail.com
c003	Cita	Depok	.75	cita@gmail.com
c004	Dika	Tangerang	-	dika@gmail.com
c005	Fifi	Depok	-	fifi@gmail.com
c006	Lili	Jakarta Selatan	.1	lili@gmail.com
c007	Melina	Jakarta Timur	.4	melina@gmail.com
c008	Nanda	Jakarta Selatan	.25	nanda@gmail.com
c009	Gina	Tangerang	1	gina@gmail.com
c010	Helmi	Jakarta Pusat	.5	helmi@gmail.com
c011	Iriana	Jakarta Selatan	.25	iriana@gmail.com
c012	Jeremy	Jakarta Barat	-	jeremy@gmail.com
c013	Kiara	Jakarta Selatan	-	kiara@gmail.com
c014	Okta	Bekasi	1.5	okta@gmail.com

2.4. RENAMING COLUMN WITH ALTER

```
ALTER TABLE ordering
RENAME COLUMN month TO order_date;
```

2.5. INSERT MULTIPLE ROWS INTO ORDERING TABLE AT ONCE (with Union All)

```
INSERT INTO ordering (ordno, order_date, quantity, customer_id, product_id)
WITH o AS(
SELECT 'ord001', TO_DATE('2022-01-01','YYYY-MM-DD'), 2, 'c010', 'p11' FROM dual UNION ALL
SELECT 'ord002', TO_DATE('2022-01-02','YYYY-MM-DD'), 4, 'c013', 'p15' FROM dual UNION ALL
SELECT 'ord003', TO_DATE('2022-01-05','YYYY-MM-DD'), 10, 'c003', 'p07' FROM dual UNION
ALL
SELECT 'ord004', TO_DATE('2022-01-12','YYYY-MM-DD'), 1, 'c004', 'p11' FROM dual UNION ALL
SELECT 'ord005', TO_DATE('2022-01-20','YYYY-MM-DD'), 5, 'c009', 'p02' FROM dual UNION ALL
SELECT 'ord006', TO_DATE('2022-02-10','YYYY-MM-DD'), 2, 'c009', 'p13' FROM dual UNION ALL
SELECT 'ord007', TO_DATE('2022-02-12','YYYY-MM-DD'), 2, 'c012', 'p12' FROM dual UNION ALL
SELECT 'ord008', TO_DATE('2022-02-14','YYYY-MM-DD'), 5, 'c008', 'p07' FROM dual UNION ALL
SELECT 'ord009', TO_DATE('2022-02-26','YYYY-MM-DD'), 3, 'c010', 'p14' FROM dual UNION ALL
SELECT 'ord010', TO_DATE('2022-03-14','YYYY-MM-DD'), 6, 'c007', 'p07' FROM dual UNION ALL
SELECT 'ord011', TO_DATE('2022-03-25','YYYY-MM-DD'), 1, 'c005', 'p05' FROM dual UNION ALL
SELECT 'ord012', TO_DATE('2022-03-29','YYYY-MM-DD'), 2, 'c001', 'p07' FROM dual UNION ALL
SELECT 'ord013', TO_DATE('2022-04-07','YYYY-MM-DD'), 3, 'c010', 'p12' FROM dual UNION ALL
SELECT 'ord014', TO_DATE('2022-04-10','YYYY-MM-DD'), 6, 'c008', 'p02' FROM dual UNION ALL
SELECT 'ord015', TO_DATE('2022-04-15','YYYY-MM-DD'), 2, 'c006', 'p01' FROM dual UNION ALL
SELECT 'ord016', TO_DATE('2022-04-18','YYYY-MM-DD'), 4, 'c007', 'p10' FROM dual UNION ALL
SELECT 'ord017', TO_DATE('2022-04-24','YYYY-MM-DD'), 1, 'c003', 'p03' FROM dual UNION ALL
SELECT 'ord018', TO_DATE('2022-04-27','YYYY-MM-DD'), 2, 'c012', 'p14' FROM dual UNION ALL
SELECT 'ord019', TO_DATE('2022-04-28','YYYY-MM-DD'), 1, 'c008', 'p13' FROM dual UNION ALL
SELECT 'ord020', TO_DATE('2022-04-30','YYYY-MM-DD'), 1, 'c009', 'p04' FROM dual UNION ALL
SELECT 'ord021', TO_DATE('2022-05-02','YYYY-MM-DD'), 1, 'c001', 'p05' FROM dual UNION ALL
SELECT 'ord022', TO_DATE('2022-05-10','YYYY-MM-DD'), 2, 'c014', 'p10' FROM dual UNION ALL
SELECT 'ord023', TO_DATE('2022-05-18','YYYY-MM-DD'), 1, 'c004', 'p14' FROM dual UNION ALL
SELECT 'ord024', TO_DATE('2022-06-24','YYYY-MM-DD'), 1, 'c002', 'p08' FROM dual UNION ALL
SELECT 'ord025', TO_DATE('2022-07-20','YYYY-MM-DD'), 1, 'c011', 'p03' FROM dual UNION ALL
SELECT 'ord026', TO_DATE('2022-07-22','YYYY-MM-DD'), 2, 'c004', 'p03' FROM dual UNION ALL
SELECT 'ord027', TO_DATE('2022-07-26','YYYY-MM-DD'), 1, 'c011', 'p03' FROM dual UNION ALL
SELECT 'ord028', TO_DATE('2022-07-27','YYYY-MM-DD'), 2, 'c003', 'p08' FROM dual UNION ALL
SELECT 'ord029', TO_DATE('2022-07-30','YYYY-MM-DD'), 1, 'c009', 'p04' FROM dual UNION ALL
SELECT 'ord030', TO_DATE('2022-08-05','YYYY-MM-DD'), 2, 'c014', 'p09' FROM dual)
SELECT * FROM o;
```

SELECT * FROM ordering

Table 7. All Data in Ordering Table

ORDNO	ORDER_DATE	QUANTITY	CUSTOMER_ID	PRODUCT_ID
ord001	01-Jan-2022	2	c010	p11
ord002	02-Jan-2022	4	c013	p15
ord003	05-Jan-2022	10	c003	p07
ord004	12-Jan-2022	1	c004	p11
ord005	20-Jan-2022	5	c009	p02
ord006	10-Feb-2022	2	c009	p13
ord007	12-Feb-2022	2	c012	p12
ord008	14-Feb-2022	5	c008	p07
ord009	26-Feb-2022	3	c010	p14
ord010	14-Mar-2022	6	c007	p07
ord011	25-Mar-2022	1	c005	p05
ord012	29-Mar-2022	2	c001	p07
ord013	07-Apr-2022	3	c010	p12
ord014	10-Apr-2022	6	c008	p02
ord015	15-Apr-2022	2	c006	p01

3. SELECT, WHERE, ORDER BY

```
SELECT c_name, c_email, c_city
FROM customer
WHERE c_city = 'Jakarta Selatan'
Order By c_name;
```

Table 8. Results of Query Select, Where, Order By

C_NAME	C_EMAIL	C_CITY
Bagus	bagus@gmail.com	Jakarta Selatan
Iriana	iriana@gmail.com	Jakarta Selatan
Kiara	kiara@gmail.com	Jakarta Selatan
Lili	lili@gmail.com	Jakarta Selatan
Nanda	nanda@gmail.com	Jakarta Selatan

4. CASE MANIPULATION (lower, upper, initcap)

SELECT LOWER(c_name), UPPER(c_name), INITCAP(c_name) FROM customer;

Table 9. Results of Query Using Case Manipulation Function

LOWER(C_NAME)	UPPER(C_NAME)	INITCAP(C_NAME)
anisa	ANISA	Anisa
bagus	BAGUS	Bagus
cita	CITA	Cita
dika	DIKA	Dika
fifi	FIFI	Fifi

5. CHARACTER MANIPULATION

5.1. CONCAT

SELECT customer_id||' - '||c_name AS username FROM customer;

Table 10. Results of Query Using Character Manipulation Function: CONCAT

USERNAME
c001 - Anisa
c002 - Bagus
c003 - Cita

5.2. SUBSTR

SELECT SUBSTR(c_name, 2, 5) FROM customer;

Table 11. Results of Query Using Character Manipulation Function: SUBSTR

SUBSTR(C_NAME,2,5)
nisa
agus
ita

5.3. LENGTH

SELECT p_name, LENGTH(p_name) FROM product;

Table 12. Results of Query Using Character Manipulation Function: LENGTH

P_NAME	LENGTH(P_NAME)
pants	5
t-shirt	7
shoes	5

5.4. INSTR

```
SELECT INSTR('c007 - Melina', 'Mel') AS "Position Found" FROM DUAL;
```

Table 13. Results of Query Using Character Manipulation Function: INSTR

Position Found
8

5.5. LPAD, RPAD

```
SELECT LPAD(c_name, 10, '*'), RPAD(c_name, 10, '*') FROM customer;
```

Table 14. Results of Query Using Character Manipulation Function: LPAD, RPAD

LPAD(C_NAME,10,'*')	RPAD(C_NAME,10,'*')
*****Anisa	Anisa*****
*****Bagus	Bagus*****
******Cita	Cita******
******Dika	Dika******

5.6. TRIM

```
SELECT TRIM('   Anisa   '),
TRIM(LEADING 'c' FROM 'c001'),
TRIM(TRAILING '1' FROM 'c001') FROM DUAL;
```

Table 15. Result of Query Using Character Manipulation Function: TRIM

TRIM('ANISA')	TRIM(LEADING'C'FROM'C001')	TRIM(TRAILING'1'FROM'C001')
Anisa	001	c00

5.7. REPLACE

```
SELECT ordno, REPLACE(ordno, '0'),
REPLACE(ordno, '0', '*') FROM ordering;
```

Table 16. Results of Query Using Character Manipulation Function: REPLACE

ORDNO	REPLACE(ORDNO,'0')	REPLACE(ORDNO,'0','*')
ord001	ord1	ord**1
ord002	ord2	ord**2
ord003	ord3	ord**3
ord004	ord4	ord**4
ord005	ord5	ord**5

6. NUMBER FUNCTIONS (round, trunc, mod)

```
SELECT price,
ROUND(price), TRUNC(price), MOD(price, 1)
FROM product;
```

Table 17. Results of Query Using Number Function: ROUND, TRUNC, MOD

PRICE	ROUND(PRICE)	TRUNC(PRICE)	MOD(PRICE,1)
30	30	30	0
15.5	16	15	.5
35	35	35	0
50	50	50	0
70.75	71	70	.75

7. DATE FUNCTIONS

7.1. ROUND, TRUNC

```
SELECT order_date, ROUND(TO_DATE(order_date),'MONTH') "Round Order Date",
TRUNC(TO_DATE(order_date),'MONTH') "Trunc Order Date" FROM ordering;
```

Table 18. Results of Query Using Date Function: ROUND, TRUNC

ORDER_DATE	Round Order Date	Trunc Order Date
20-Jan-2022	01-Feb-2022	01-Jan-2022
10-Feb-2022	01-Feb-2022	01-Feb-2022
12-Feb-2022	01-Feb-2022	01-Feb-2022
14-Feb-2022	01-Feb-2022	01-Feb-2022
26-Feb-2022	01-Mar-2022	01-Feb-2022

7.2. MONTHS_BETWEEN

```
SELECT ordno, product_id, quantity, customer_id, ROUND(MONTHS_BETWEEN(sysdate,
order_date))||' Months Ago' "ORDERED" FROM ordering;
```

Table 19. Results of Query Using Date Function: MONTHS_BETWEEN

ORDNO	PRODUCT_ID	QUANTITY	CUSTOMER_ID	ORDERED
ord001	p11	2	c010	8 Months Ago
ord002	p15	4	c013	8 Months Ago
ord003	p07	10	c003	7 Months Ago
ord004	p11	1	c004	7 Months Ago
ord005	p02	5	c009	7 Months Ago

7.3. ADD_MONTHS

SELECT SYSDATE, ADD_MONTHS(SYSDATE, 6) FROM DUAL;

Table 20. Results of Query Using Date Function: ADD_MONTHS

SYSDATE	ADD_MONTHS(SYSDATE,6)
18-Aug-2022	18-Feb-2023

7.4. NEXT_DAY

SELECT SYSDATE, NEXT_DAY(SYSDATE, 'saturday') "NEXT WEEKEND" FROM DUAL;

Table 21. Results of Query Using Date Function: NEXT_DAY

SYSDATE	NEXT WEEKEND
18-Aug-2022	20-Aug-2022

7.5. LAST_DAY

SELECT SYSDATE, LAST_DAY(SYSDATE) FROM DUAL;

Table 22. Results of Query Using Date Function: LAST_DAY

SYSDATE	LAST_DAY(SYSDATE)
18-Aug-2022	31-Aug-2022

8. CONVERSION FUNCTIONS

8.1. NUMBER TO CHARACTER

SELECT price, TO_CHAR(price, '$9999.9') FROM product;

Table 23. Results of Query Using Convertion Function: NUMBER TO_CHAR

PRICE	TO_CHAR(PRICE,'$9999.9')
30	$30.0
15.5	$15.5
35	$35.0
50	$50.0
70.75	$70.8

8.2. CHARACTER TO NUMBER

SELECT TO_NUMBER('$1,250.50', '$9,999.99') FROM DUAL;

Table 24. Results of Query Using Convertion Function: CHAR TO_NUMBER

TO_NUMBER('$1,250.50','$9,999.99')
1250.5

8.3. DATE TO CHARACTER

SELECT order_date, TO_CHAR(order_date, 'Day, Mon dd yyyy') FROM ordering;

Table 25. Results of Query Using Convertion Function: DATE TO_CHAR

ORDER_DATE	TO_CHAR(ORDER_DATE,'DAY,MONDDYYYY')
01-Jan-2022	Saturday , Jan 01 2022
02-Jan-2022	Sunday , Jan 02 2022
05-Jan-2022	Wednesday, Jan 05 2022
12-Jan-2022	Wednesday, Jan 12 2022
20-Jan-2022	Thursday , Jan 20 2022

8.4. CHARACTER TO DATE

SELECT TO_DATE('August 17, 2022', 'Month dd, yyyy') FROM DUAL;

Table 26. Result of Query Using Convertion Function: CHAR TO_DATE

TO_DATE('AUGUST17,2022','MONTHDD,YYYY')
17-Aug-2022

9. NULL FUNCTIONS

9.1. NVL

SELECT c_name, discount, TO_CHAR(NVL (discount, 0), '$99.99') FROM customer;

Table 27. Results of Query Using NULL Function: NVL

C_NAME	DISCOUNT	TO_CHAR(NVL(DISCOUNT,0),'$99.99')
Anisa	-	$.00
Bagus	-	$.00
Cita	.75	$.75
Dika	-	$.00
Fifi	-	$.00

9.2. NVL2

SELECT c_name, discount, NVL2 (discount, 'With discount', 'Without Discount') AS "DISCOUNT" FROM customer;

Table 28. Results of Query Using NULL Function: NVL2

C_NAME	DISCOUNT	DISCOUNT
Anisa	-	Without Discount
Bagus	-	Without Discount
Cita	.75	With discount
Dika	-	Without Discount
Fifi	-	Without Discount

9.3. NULLIF

--Will return the length of customer_id (as the first value specified in the parentheses) if the length of customer_id and product_id is not equal. If the length of both values specified in the parentheses are equal, nullif will return NULL.

SELECT NULLIF(LENGTH(customer_id), LENGTH(product_id)), NULLIF(0.25, 0.25) FROM ordering;

Table 29. Results of Query Using NULL Function: NULLIF

NULLIF(LENGTH(CUSTOMER_ID),LENGTH(PRODUCT_ID))	NULLIF(.25,.25)
4	-
4	-
4	-
4	-
4	-

9.4. COALESCE

--Coalesce will return the first not null value in the parentheses
SELECT COALESCE(NULL, NULL, NULL, 'EXAMPLE', NULL, 'ANOTHER EXAMPLE') FROM DUAL;

Table 30. Result of Query Using NULL Function: COALESCE

COALESCE(NULL,NULL,NULL,'EXAMPLE',NULL,'ANOTHEREXAMPLE')
EXAMPLE

10. CONDITIONAL EXPRESSION

10.1. DECODE

SELECT ordno, DECODE(mod(quantity, 2), 1, 'Odd', 0, 'Even') FROM ordering;

Table 31. Results of Query Using Conditional Expression: Decode

ORDNO	DECODE(MOD(QUANTITY,2),1,'ODD',0,'EVEN')
ord001	Even
ord002	Even
ord003	Even
ord004	Odd

10.2. CASE

```
SELECT customer_id, CASE
  WHEN quantity < 3 THEN 'low'
  WHEN quantity > 3 AND quantity <= 5 THEN 'average'
  WHEN quantity > 5 THEN 'high'
  END order_amount
FROM ordering;
```

Table 32. Results of Query Using Conditional Expression: Case

CUSTOMER_ID	ORDER_AMOUNT
c010	low
c013	average
c003	high
c004	low
c009	average

11. JOIN PART I

11.1. CROSS JOIN

--Cross join produces cartesian product which returns every possible combinations of each row in table --1 to each row in table 2

```
SELECT customer_id, ordering.product_id, ordering.quantity, order_date
FROM product CROSS JOIN ordering
ORDER BY customer_id;

--This query will return 450 rows
```

Table 33. Results of Query Using Cross Join

Results | Explain | Describe | Saved SQL | History

c014	p09	2	05-Aug-2022
c014	p10	2	10-May-2022
c014	p09	2	05-Aug-2022
c014	p10	2	10-May-2022
c014	p09	2	05-Aug-2022
c014	p10	2	10-May-2022

450 rows returned in 0.02 seconds Download

11.2. NATURAL JOIN, JOIN ON, JOIN USING, INNER JOIN

```
--This query will return 30 rows
SELECT * FROM customer NATURAL JOIN ordering;
```

Table 34. Results of Query Using Natural Join

CUSTOMER_ID	C_NAME	C_CITY	DISCOUNT	C_EMAIL	ORDNO
c010	Helmi	Jakarta Pusat	.5	helmi@gmail.com	ord001
c013	Kiara	Jakarta Selatan	-	kiara@gmail.com	ord002
c003	Cita	Depok	.75	cita@gmail.com	ord003
c004	Dika	Tangerang	-	dika@gmail.com	ord004

11.3. INNER JOIN WITH ON/USING

```
SELECT p.p_name, o.quantity, p.price, o.customer_id
FROM product p INNER JOIN ordering o
ON (p.product_id = o.product_id);

SELECT p.p_name, o.quantity, p.price, o.customer_id
FROM product p INNER JOIN ordering o
USING (product_id);
--Both of these queries will return the same results
```

Table 35. Results of Query Using Inner Join with ON or USING

P_NAME	QUANTITY	PRICE	CUSTOMER_ID
hat	2	10.99	c010
sunglass	4	40	c013
mask	10	1.5	c003
hat	1	10.99	c004
t-shirt	5	15.5	c009

11.4. SELF JOIN

```
SELECT DISTINCT p.p_name, p.price, a.customer_id "1stBuyer",
b.customer_id "2ndBuyer",  a.quantity AS "1stOrder", b.quantity AS "2ndOrder"
FROM ordering a INNER JOIN ordering b
ON(a.customer_id<>b.customer_id
AND a.product_id = b.product_id
AND a.order_date <> b.order_date)
INNER JOIN product p
ON(b.product_id=p.product_id)
ORDER BY a.customer_id;
```

Table 36. Results of Query Using Inner Self Join

P_NAME	PRICE	1stBuyer	2ndBuyer	1stOrder	2ndOrder
mask	1.5	c001	c003	2	10
mask	1.5	c001	c007	2	6
mask	1.5	c001	c008	2	5
glasses	70.75	c001	c005	1	1
bag	25	c002	c003	1	2

11.5. LEFT OUTER JOIN

```
SELECT p.product_id, p.p_name, p.price,  o.quantity, o.order_date
FROM product p LEFT JOIN ordering o
ON(p.product_id = o.product_id)
WHERE price < 10
ORDER BY p.product_id;
```

Table 37. Results of Query Using Left Outer Join

PRODUCT_ID	P_NAME	PRICE	QUANTITY	ORDER_DATE
p06	brecelet	5	-	-
p07	mask	1.5	10	05-Jan-2022
p07	mask	1.5	5	14-Feb-2022
p07	mask	1.5	6	14-Mar-2022
p07	mask	1.5	2	29-Mar-2022

11.6. RIGHT OUTER JOIN

```
SELECT p.product_id, p.p_name, p.price,  o.quantity, o.order_date
FROM product p RIGHT JOIN ordering o
ON(p.product_id = o.product_id)
WHERE price < 10
ORDER BY p.product_id;
```

Table 38. Results of Query Using Right Outer Join

PRODUCT_ID	P_NAME	PRICE	QUANTITY	ORDER_DATE
p07	mask	1.5	10	05-Jan-2022
p07	mask	1.5	5	14-Feb-2022
p07	mask	1.5	6	14-Mar-2022
p07	mask	1.5	2	29-Mar-2022

11.7. FULL OUTER JOIN

```
SELECT p.product_id, p.p_name, p.price,  o.quantity, o.order_date
FROM product p FULL OUTER JOIN ordering o
ON(p.product_id = o.product_id)
WHERE price < 10
ORDER BY p.product_id;
```

Table 39. Results of Query Using Full Outer Join

PRODUCT_ID	P_NAME	PRICE	QUANTITY	ORDER_DATE
p06	brecelet	5	-	-
p07	mask	1.5	10	05-Jan-2022
p07	mask	1.5	5	14-Feb-2022
p07	mask	1.5	6	14-Mar-2022
p07	mask	1.5	2	29-Mar-2022

12. JOINS PART II

12.1. EQUI JOIN

```
SELECT p.product_id, p.p_name, p.price, o.quantity, p.quantity, o.order_date
FROM product p INNER JOIN ordering o
ON(o.quantity = p.quantity)
ORDER BY p.product_id;
```

Table 40. Equi Join Example

PRODUCT_ID	P_NAME	PRICE	QUANTITY	QUANTITY	ORDER_DATE
p02	t-shirt	15.5	10	10	05-Jan-2022
p04	jacket	50	5	5	20-Jan-2022
p04	jacket	50	5	5	14-Feb-2022
p12	shorts	20.25	5	5	20-Jan-2022
p12	shorts	20.25	5	5	14-Feb-2022
p14	earphone	30	10	10	05-Jan-2022

12.2. NON EQUI JOIN

```
SELECT p.product_id, p.p_name, p.price, o.quantity, p.quantity, o.order_date
FROM product p INNER JOIN ordering o
ON(o.quantity < p.quantity)
WHERE order_date = '01-Jan-2022'
ORDER BY p.product_id;
```

Table 41. Non Equi Join Example

PRODUCT_ID	P_NAME	PRICE	ORDER_QUANTITY	PRODUCT_QUANTITY	ORDER_DATE
p01	pants	30	2	20	01-Jan-2022
p02	t-shirt	15.5	2	10	01-Jan-2022
p03	shoes	35	2	12	01-Jan-2022
p04	jacket	50	2	5	01-Jan-2022
p05	glasses	70.75	2	25	01-Jan-2022
p06	brecelet	5	2	50	01-Jan-2022
p07	mask	1.5	2	120	01-Jan-2022
p08	bag	25	2	12	01-Jan-2022
p09	headphone	60	2	30	01-Jan-2022
p10	shirt	10.15	2	50	01-Jan-2022
p11	hat	10.99	2	20	01-Jan-2022
p12	shorts	20.25	2	5	01-Jan-2022
p13	earrings	35.5	2	60	01-Jan-2022
p14	earphone	30	2	10	01-Jan-2022
p15	sunglass	40	2	20	01-Jan-2022

13. GROUP FUNCTION

13.1. SUM

--Wants to know which types of product are the most ordered during 2 weeks before Eid Mubarak

```
SELECT c.c_name, p.p_name, SUM(o.quantity) AS Total_Ordered
FROM ordering o INNER JOIN product p
ON(o.product_id = p.product_id)
INNER JOIN customer c ON (o.customer_id = c.customer_id)
WHERE o.order_date BETWEEN '15-April-2022' AND '01-May-2022'
GROUP BY o.product_id, o.customer_id, p.p_name, c.c_name
ORDER BY Total_Ordered DESC;
```

Table 42. Results of Query Using Group Function: SUM

C_NAME	P_NAME	TOTAL_ORDERED
Melina	shirt	4
Lili	pants	2
Jeremy	earphone	2
Nanda	earrings	1
Cita	shoes	1

13.2. COUNT

```
SELECT COUNT(DISTINCT p.product_id) "Types of Products Available", COUNT(DISTINCT
o.product_id) "Types of Products Ordered"
FROM product p CROSS JOIN ordering o;
```

Table 43. Results of Query Using Group Function: COUNT

Types of Products Available	Types of Products Ordered
15	14

13.3. NVL, AVG

```
SELECT TO_CHAR(AVG(NVL(discount, 0)), '$99.99') AS "AVERAGE_DISCOUNT"
FROM customer;
```

Table 44. Results of Query Using Group Function: NVL, AVG

AVERAGE_DISCOUNT
$.34

13.4. HAVING CLAUSE

--Wants to know the biggest quantity of order for each product type, with special condition: each product must had been ordered by at least more than one customers

```
SELECT product_id, MAX(quantity) AS MAX_QUANTITY
FROM ordering
GROUP BY product_id
HAVING COUNT(customer_id)>1
ORDER BY MAX_QUANTITY DESC;
```

Table 45. Results of Query Using Group Function: HAVING CLAUSE

PRODUCT_ID	MAX_QUANTITY
p07	10
p02	6
p10	4
p12	3
p14	3

14. GROUP FUNCTIONS PART II

14.1. ROLLUP

```
SELECT customer_id, product_id, SUM(quantity)
FROM ordering
GROUP BY ROLLUP (customer_id, product_id)
WHERE customer_id IN('c012', 'c013',  'c014');
```

Table 46. Results of Query Using Group Function: ROLLUP

CUSTOMER_ID	PRODUCT_ID	SUM(QUANTITY)
c012	p12	2
c012	p14	2
c012	-	4
c013	p15	4
c013	-	4
c014	p09	2
c014	p10	2
c014	-	4
-	-	12

14.2. CUBE

```
SELECT customer_id, product_id, SUM(quantity)
FROM ordering
WHERE customer_id IN('c012', 'c013',  'c014')
```

Table 47. Results of Query Using Group Function: CUBE

CUSTOMER_ID	PRODUCT_ID	SUM(QUANTITY)
c012	p12	2
c012	p14	2
c012	-	4
c013	p15	4
c013	-	4
c014	p09	2
c014	p10	2
c014	-	4
-	-	12

14.3. GROUPING SETS

```
SELECT ordno, customer_id, product_id, SUM(quantity)
FROM ordering
WHERE customer_id IN('c012', 'c013',  'c014')
GROUP BY GROUPING SETS ((ordno, product_id),(customer_id, product_id),(customer_id,
ordno));
```

Table 48. Results of Query Using Group Function: GROUPING SETS

ORDNO	CUSTOMER_ID	PRODUCT_ID	SUM(QUANTITY)
ord030	-	p09	2
ord007	-	p12	2
ord022	-	p10	2
ord002	-	p15	4
ord018	-	p14	2
-	c013	p15	4
-	c012	p14	2
-	c012	p12	2
-	c014	p09	2
-	c014	p10	2
ord007	c012	-	2
ord030	c014	-	2
ord002	c013	-	4
ord018	c012	-	2
ord022	c014	-	2

14.4. GROUPING FUNCTIONS

--grouping functions --> 1 for computed row, 0 for non-computed row

```
SELECT customer_id, product_id, SUM(quantity),
GROUPING(customer_id) AS "Customer Sub Total",
GROUPING(product_id) AS "Product Sub Total"
FROM ordering
WHERE customer_id IN('c012', 'c013',  'c014')
GROUP BY CUBE (customer_id, product_id);
```

Table 49. Results of Query Using Group Function: GROUPING FUNCTIONS

CUSTOMER_ID	PRODUCT_ID	SUM(QUANTITY)	Customer Sub Total	Product Sub Total
-	-	12	1	1
-	p09	2	1	0
-	p10	2	1	0
-	p12	2	1	0
-	p14	2	1	0
-	p15	4	1	0
c012	-	4	0	1
c012	p12	2	0	0
c012	p14	2	0	0
c013	-	4	0	1
c013	p15	4	0	0
c014	-	4	0	1
c014	p09	2	0	0
c014	p10	2	0	0

15. SUBQUERIES

15.1. SINGLE ROW SUBQUERIES (FROM DIFFERENT TABLES)

```
SELECT customer_id, quantity, product_id
FROM ordering
WHERE customer_id =
(SELECT customer_id FROM customer WHERE c_name = 'Fifi');
```

Table 50. Results of Subqueries: SINGLE ROW

CUSTOMER_ID	QUANTITY	PRODUCT_ID
c005	1	p05

15.2. MULTIPLE ROW SUBQUERIES

```
SELECT customer_id, order_date, product_id, quantity
FROM ordering
WHERE(product_id,customer_id) IN
(SELECT product_id, customer_id FROM ordering WHERE quantity IN(2, 3));
```

Table 51. Results of Subqueries: MULTIPLE ROW

CUSTOMER_ID	ORDER_DATE	PRODUCT_ID	QUANTITY
c013	02-Jan-2022	p15	4
c010	26-Feb-2022	p14	3
c010	07-Apr-2022	p12	3
c007	18-Apr-2022	p10	4

15.3. CORRELATED SUBQUERIES

```
WITH products AS
(SELECT DISTINCT product_id FROM ordering WHERE product_id IS NOT NULL)
SELECT p_name AS "Never ordered product"
FROM product
WHERE product_id NOT IN
(SELECT * FROM products);
```

Table 52. Results of Subqueries: CORRELATED

Never ordered product
brecelet

16. VIEW

16.1. CREATE VIEW

```
CREATE OR REPLACE VIEW view_products_ordered
("order_number", "customerID", "productID", "product_name", "ordered_quantity",
"product_price")
AS SELECT o.ordno, o.customer_id, o.product_id, p.p_name, o.quantity, p.price
FROM ordering o JOIN product p
ON (o.product_id= p.product_id);
```

Table 53. Create View Results

16.2. SHOW VIEW TABLE

```
SELECT * FROM view_products_ordered;

--this view is considered as complex view because it is created using join function,
therefore, we are not allowed to use data manipulation queries such as alter or update on
this complex view.
```

Table 54. Showing View Table

order_number	customerID	productID	product_name	ordered_quantity	product_price
ord001	c010	p11	hat	2	10.99
ord002	c013	p15	sunglass	4	40
ord003	c003	p07	mask	10	1.5
ord004	c004	p11	hat	1	10.99
ord005	c009	p02	t-shirt	5	15.5
ord006	c009	p13	earrings	2	35.5
ord007	c012	p12	shorts	2	20.25
ord008	c008	p07	mask	5	1.5
ord009	c010	p14	earphone	3	30
ord010	c007	p07	mask	6	1.5
ord011	c005	p05	glasses	1	70.75
ord012	c001	p07	mask	2	1.5
ord013	c010	p12	shorts	3	20.25
ord014	c008	p02	t-shirt	6	15.5
ord015	c006	p01	pants	2	30
ord016	c007	p10	shirt	4	10.15
ord017	c003	p03	shoes	1	35
ord018	c012	p14	earphone	2	30
ord019	c008	p13	earrings	1	35.5
ord020	c009	p04	jacket	1	50
ord021	c001	p05	glasses	1	70.75
ord022	c014	p10	shirt	2	10.15
ord023	c004	p14	earphone	1	30
ord024	c002	p08	bag	1	25
ord025	c011	p03	shoes	1	35
ord026	c004	p03	shoes	2	35
ord027	c011	p03	shoes	1	35
ord028	c003	p08	bag	2	25
ord029	c009	p04	jacket	1	50
ord030	c014	p09	headphone	2	60

16.3. DROP VIEW

```
DROP VIEW view_products_ordered;
```

Table 55. Drop View Results

Results	Explain	Describe	Saved SQL	History

```
View dropped.
```

16.4. TOP N ANALYSIS

```
SELECT ROWNUM as RANK, customer_id TOP_BUYER, quantity
FROM (SELECT customer_id, quantity
FROM ordering
ORDER BY quantity DESC)
WHERE ROWNUM <=5;
```

Table 56. Results of Queries Using TOP N ANALYSIS

RANK	TOP_BUYER	QUANTITY
1	c003	10
2	c007	6
3	c008	6
4	c009	5
5	c008	5

17. SEQUENCE

17.1. CREATE SEQUENCE

```
CREATE SEQUENCE supplier_id_seq
INCREMENT BY 1
START WITH 1
MAXVALUE 1000
NOCACHE
NOCYCLE;
```

Table 57. Create Sequence Results

Results	Explain	Describe	Saved SQL	History

```
Sequence created.
```

17.2. USING A SEQUENCE IN A TABLE

```
CREATE TABLE supplier
(supplier_id_seq NUMBER(4,0) CONSTRAINT supplier_id_seq_pk PRIMARY KEY,
supplier_name VARCHAR2(20),
location VARCHAR(20));
```

```
DESCRIBE TABLE supplier;
```

Table 58. Supplier Table Structure

Column Name	Data Type	Nullable	Default	Primary Key
SUPPLIER_ID_SEQ	NUMBER(4,0)	No	-	1
SUPPLIER_NAME	VARCHAR2(20)	Yes	-	-
LOCATION	VARCHAR2(20)	Yes	-	-

```
insert nextval and currval pseudocolumns into table rows
INSERT INTO supplier (supplier_id_seq, supplier_name, location)
VALUES(supplier_id_seq.CURRVAL, 'PT ABC', 'Bogor');

INSERT INTO supplier (supplier_id_seq, supplier_name, location)
VALUES(supplier_id_seq.NEXTVAL, 'PT XYZ', 'Tangerang');
```

```
SELECT * FROM supplier
```

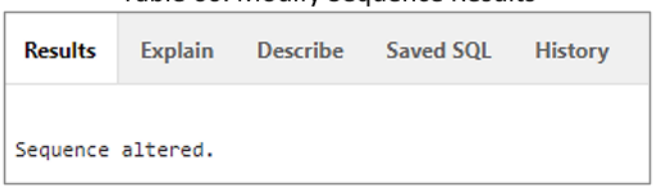
Table 59. Supplier Table Data with Sequence as Primary Key

SUPPLIER_ID_SEQ	SUPPLIER_NAME	LOCATION
1	PT ABC	Bogor
2	PT XYZ	Tangerang

17.3. MODIFYING A SEQUENCE

```
ALTER SEQUENCE supplier_id_seq
INCREMENT BY 1
MAXVALUE 500
NOCACHE
NOCYCLE;
```

Table 60. Modify Sequence Results



17.4. DROP A SEQUENCE

DROP SEQUENCE supplier_id_seq;

Table 61. Drop Sequence Results

Results	Explain	Describe	Saved SQL	History
Sequence dropped.				

18. INDEX

18.1. CREATE INDEX

CREATE INDEX customer_name_idx ON customer(c_name);

Table 62. Create Index Results

Results	Explain	Describe	Saved SQL	History
Index created.				

18.2. DROP INDEX

DROP INDEX customer_name_idx;

Table 63. Drop Index Results

Results	Explain	Describe	Saved SQL	History
Index dropped.				

19. SYNONYM

19.1. CREATE SYNONYM

CREATE SYNONYM buying FOR ordering;

Table 64. Create Synonym Results

Results	Explain	Describe	Saved SQL	History
Synonym created.				

SELECT * FROM buying
--Results will show the ordering table as previously shown on table 6

19.2. DROP SYNONIM

DROP SYNONYM buying

Table 65. Drop Synonym Results

Results	Explain	Describe	Saved SQL	History
Synonym dropped.				

REFERENSI

Oracle. (2018). *Oracle Live SQL - Script: Inserting Multiple Rows Using a Single Statement*. Retrieved August 19, 2022, from Live SQL: https://livesql.oracle.com/apex/livesql/file/content_BM1LJQ87M5CNIOKPOWPV6ZGR3.html

Oracle Academy. (2022). *DP Database Programming with SQL Learner - English*. Retrieved July 11, 2022, from Oracle Academy: https://myacademy.oracle.com/lmt/clmslearningpathdetails.prmain?in_sessionId=1A34J939AJA98440&in_learningPathId=72450131&in_from_module=LMTLOGIN.PRMENU

TechOnTheNet.com. (2003-2022). *Oracle / PLSQL: REPLACE Function*. Retrieved August 10, 2022, from TechOnTheNet.com: https://www.techonthenet.com/oracle/functions/replace.php

TechOnTheNet.com. (2003-2022). *Oracle / PLSQL: VIEW*. Retrieved August 20, 2022, from TechOnTheNet.com: https://www.techonthenet.com/oracle/views.php#:~:text=The%20syntax%20for%20the%20CREATE,that%20you%20wish%20to%20create.