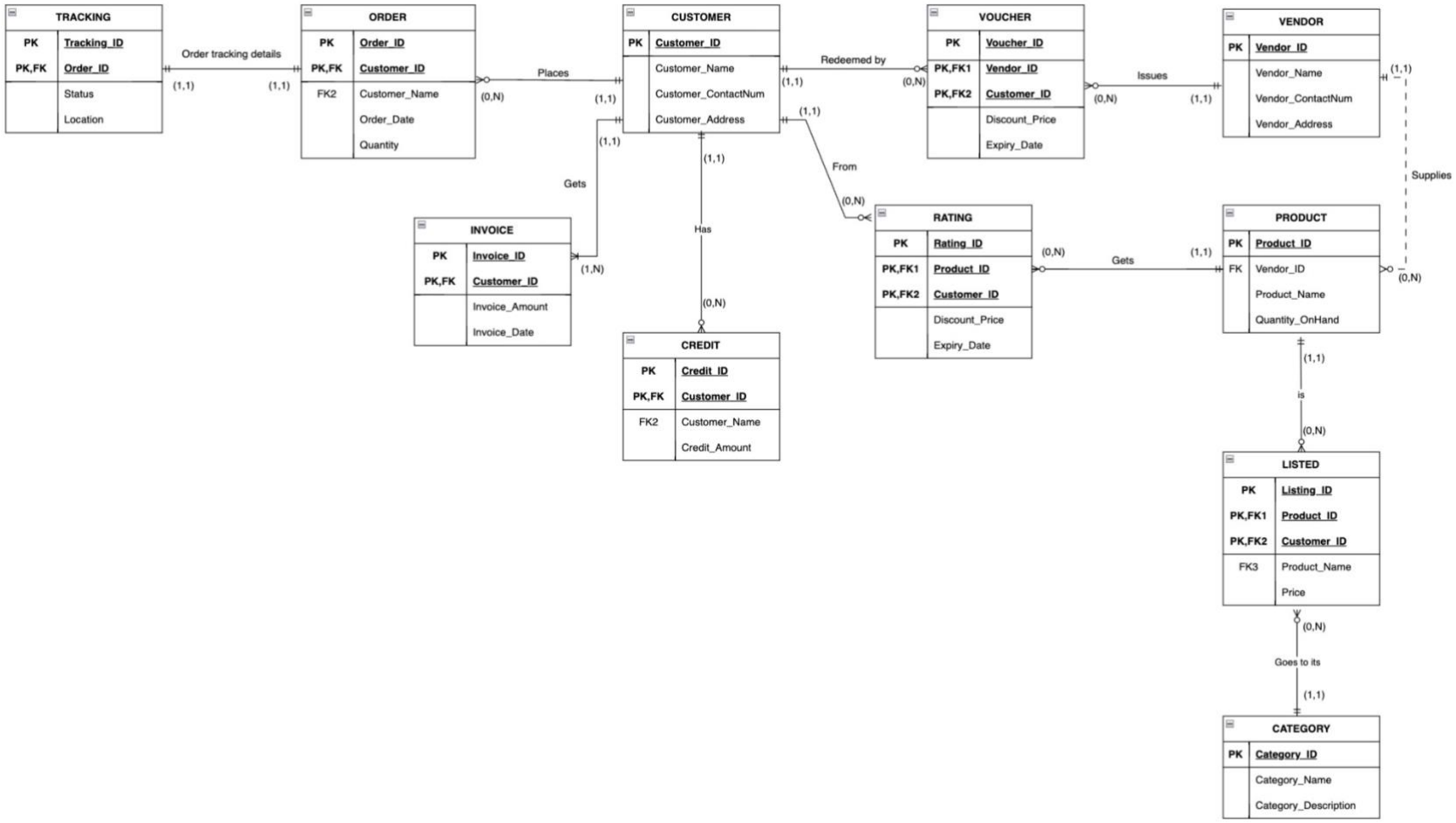
TIS1101 Database Fundamentals
Assignment 2

Title: Lazada Malaysia

No.	Student ID	Name	Role	Email
1.	1221304243	Rayyan	**Leader**	1221304243@student.mmu.edu.my
2.	1221304617	Mostafa Kamel	Member	1221304617@student.mmu.edu.my
3.	1221301252	Alkadi, Mohamad Nour	Member	1221301252@student.mmu.edu.my
4.	1211201716	Aiman Hareez bin Haamirisyuan Aidil	Member	1211201716@student.mmu.edu.my

ERD DIAGRAM


TRACKING
PK Tracking_ID
PK,FK Order_ID
Status
Location
Order tracking details
(1,1)
(1,1)
ORDER
PK Order_ID
PK,FK Customer_ID
FK2 Customer_Name
Order_Date
Quantity
(0,N)
Places
(1,1)
CUSTOMER
PK Customer_ID
Customer_Name
Customer_ContactNum
Customer_Address
(1,1)
Redeemed by
(1,1)
(0,N)
VOUCHER
PK Voucher_ID
PK,FK1 Vendor_ID
PK,FK2 Customer_ID
Discount_Price
Expiry_Date
(0,N)
Issues
(1,1)
VENDOR
PK Vendor_ID
Vendor_Name
Vendor_ContactNum
Vendor_Address
(1,1)
Supplies
(0,N)
(1,1)
Gets
(1,N)
INVOICE
PK Invoice_ID
PK,FK Customer_ID
Invoice_Amount
Invoice_Date
(1,1)
Has
(0,N)
CREDIT
PK Credit_ID
PK,FK Customer_ID
FK2 Customer_Name
Credit_Amount
(1,1)
From
(0,N)
RATING
PK Rating_ID
PK,FK1 Product_ID
PK,FK2 Customer_ID
Discount_Price
Expiry_Date
(0,N)
Gets
(1,1)
PRODUCT
PK Product_ID
FK Vendor_ID
Product_Name
Quantity_OnHand
(1,1)
is
(0,N)
LISTED
PK Listing_ID
PK,FK1 Product_ID
PK,FK2 Customer_ID
FK3 Product_Name
Price
(0,N)
Goes to its
(1,1)
CATEGORY
PK Category_ID
Category_Name
Category_Description

1. Data Dictionary

Changes in erd attributes:
Rating entity- removed Discount_price,expiry_date, added Rating_Amount
Listed entity- Category_ID instead of Customer_ID

TABLE NAME	ATTRIBUTE NAME	DATA TYPE	FORMAT	Null/Not Null	PK OR FK	Description
CUSTOMER	Customer_ID	CHAR(4)	9999	NOT NULL	PK	Holds unique student ids.
	Customer_Name	VARCHAR(20)	xxxxxxxxxxxxxx			Stores student's name.
	Customer_ContactNum	VARCHAR(11)	99999999999			Stores customers contact num.
	Customer_Address	VARCHAR(40)	xxxxxxxxxxxxxx			Stores customer's address
ORDER	Order_ID	CHAR(7)	ORD9999	NOT NULL	PK	Holds Unique order ids.
	Customer_ID	CHAR(4)	9999	NOT NULL	PK, FK	Foreign key referencing customer table.
	Customer_Name	VARCHAR(20)	xxxxxxxxxxxxxx		FK	Foreign key referencing customer table.
	Order_Date	DATE	YYYY-MM-DD			Stores date the order was placed.
	Quantity	INT	999			Stores quantity of order.
TRACKING	Tracking_ID	CHAR(7)	TRC9999	NOT NULL	PK	Holds unique tracking ids.
	Order_ID	CHAR(7)	ORD9999	NOT NULL	PK, FK	Foreign key referencing order table.
	Status	Varchar(15)	xxxxxxxxxxxxx			Stores order status(delivered or not delivered)

	Location	VARCHAR(40)	xxxxxxxxxxxxx			Stores current order location.
INVOICE	Invoice_ID	CHAR(7)	INV9999	NOT NULL	PK	Holds unique invoice ids
	Customer_ID	CHAR(4)	9999	NOT NULL	PK, FK	Foreign key referencing customer table.
	Invoice_Amount	DECIMAL(7, 2)	99999.99			Holds invoice amount
	Invoice_Date	DATE	YYYY-MM-DD			Holds invoice date
CREDIT	Credit_ID	CHAR(7)	CRD9999	NOT NULL	PK	Holds unique credit ids
	Customer_ID	CHAR(4)	9999	NOT NULL	PK, FK	Foreign key referencing customer table
	Customer_Name	VARCHAR(20)	xxxxxxxxxxxxx		FK	Stores customers names
	Credit_Amount	Decimal(7,2)	999999,99			Store credit amount.
RATING	Rating_ID	CHAR(7)	RTN9999	NOT NULL	PK	Holds credit id
	Rating_Amount	INT	0-5			Stores rating(0-5)
	Product_ID	CHAR(7)	PRD9999	NOT NULL	PK, FK	Foreign key referencing product table
	Customer_ID	CHAR(4)	9999	NOT NULL	PK, FK	Foreign key referencing customer table
VOUCHER	Voucher_ID	CHAR(7)	VCH9999	NOT NULL	PK	Holds unique voucher ids
	Vendor_ID	CHAR(7)	VND9999	NOT NULL	PK, FK	Foreign key referencing vendor table
	Customer_ID	CHAR(4)	9999	NOT NULL	PK, FK	Foreign key referencing customer table

	Discount_Price	INT	9999			Holds discount price.
	Expiry_Date	Date	YYYY-MM-DD			Stores vouchers expiry date
VENDOR	Vendor_ID	CHAR(7)	VND9999	NOT NULL	PK	Holds unique vendor ids
	Vendor_Name	VARCHAR(20)	xxxxxxxxxxxxxx			Stores vendor names
	Vendor_ContactNum	VARCHAR(13)	99999999999			Stores vendors contact num.
	Vendor_Address	VARCHAR(40)	xxxxxxxxxxxxxx			Stores vendors address
PRODUCT	Product_ID	CHAR(7)	PRD9999	NOT NULL	PK	Holds unique product ids

	Vendor_ID	CHAR(7)	VND9999	NOT NULL	FK	Foreign key referencing vendor table
	Product_Name	VARCHAR(30)	xxxxxxxxxxxxxx			Stores product names
	Quantity_OnHand	INT	999			Holds product quantity available
LISTED	Listing_ID	CHAR(7)	LST9999	NOT NULL	PK	Stores unique listing ids
	Product_ID	CHAR(7)	PRD9999	NOT NULL	PK, FK	Foreign key referencing product table.
	Category_ID	CHAR(4)	9999	NOT NULL	PK, FK	Foreign key referencing category table.
	Product_Name	VARCHAR(20)	xxxxxxxxxxxxxx		FK	Foreign key referencing
	Price	DECIMAL(7, 2)	99999.99			Holds price of product listed in category

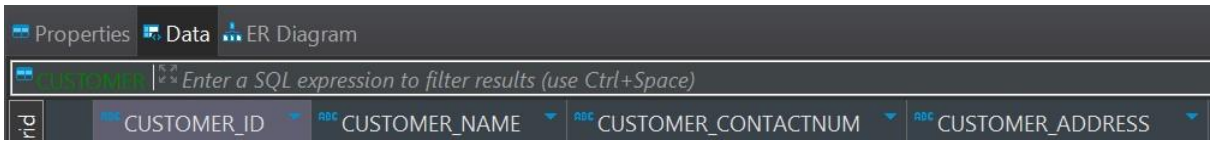
CATEGORY	Category_ID	CHAR(7)	CAT9999	NOT NULL	PK	Holds unique category ids
	Category_Name	VARCHAR(20)	xxxxxxxxxxxxx			Stores category name
	Category_Description	VARCHAR(150)	xxxxxxxxxxxxx			Stores category description

2. Creation of Tables

Customer Table

```
CREATE TABLE Customer (
CUSTOMER_ID             CHAR(4) NOT NULL PRIMARY KEY,
CUSTOMER_NAME           VARCHAR(20),
CUSTOMER_CONTACTNUM         VARCHAR(11),
CUSTOMER_ADDRESS        VARCHAR(40))
```

Column name	Data type schema	Data type name	Column Length	Scale	Nulls
CUSTOMER_ID	SYSIBM	CHARACTER	4	0	No
CUSTOMER_NAME	SYSIBM	VARCHAR	20	0	Yes
CUSTOMER_CONTACTNUM	SYSIBM	VARCHAR	11	0	Yes
CUSTOMER_ADDRESS	SYSIBM	VARCHAR	40	0	Yes



Order Table

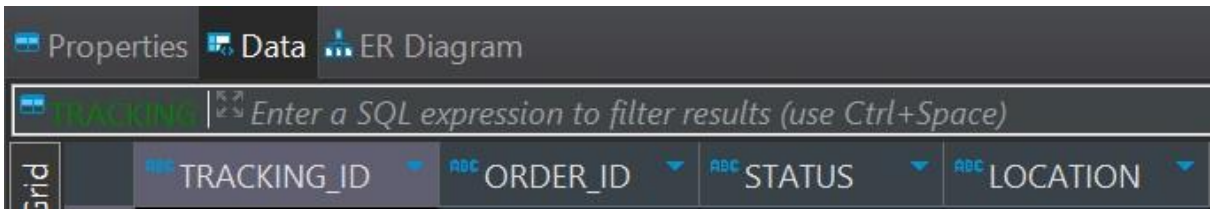
```
CREATE TABLE ORDER (
ORDER_ID                CHAR(7) NOT NULL PRIMARY KEY,
CUSTOMER_ID             CHAR(4) NOT NULL,
CUSTOMER_NAME           VARCHAR(20),
ORDER_DATE                  DATE,
QUANTITY                INT,
FOREIGN KEY (CUSTOMER_ID) REFERENCES CUSTOMER,
FOREIGN KEY (CUSTOMER_NAME) REFERENCES CUSTOMER)
```

Column name	Data type schema	Data type name	Column Length	Scale	Nulls
ORDER_ID	SYSIBM	CHARACTER	7	0	No
CUSTOMER_ID	SYSIBM	CHARACTER	4	0	Yes
CUSTOMER_NAME	SYSIBM	VARCHAR	20	0	Yes
ORDER_DATE	SYSIBM	DATE	4	0	Yes
QUANTITY	SYSIBM	INTEGER	4	0	Yes

Tracking Table

```
CREATE TABLE TRACKING (
TRACKING_ID                    CHAR(7) NOT NULL PRIMARY KEY,
ORDER_ID                       CHAR(7) NOT NULL,
STATUS                         VARCHAR(20),
LOCATION                       VARCHAR(40),
FOREIGN KEY (ORDER_ID) REFERENCES ORDER)
```

Column name	Data type schema	Data type name	Column Length	Scale	Nulls
TRACKING_ID	SYSIBM	CHARACTER	7	0	No
ORDER_ID	SYSIBM	CHARACTER	7	0	Yes
STATUS	SYSIBM	VARCHAR	20	0	Yes
LOCATION	SYSIBM	VARCHAR	40	0	Yes



Invoice Table

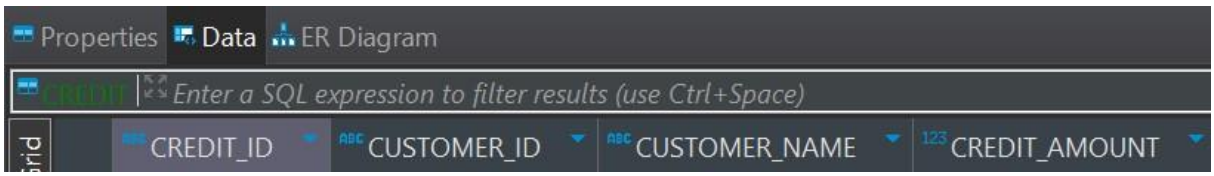
```
CREATE TABLE INVOICE (
INVOICE_ID                     CHAR(7) NOT NULL PRIMARY KEY,
CUSTOMER_ID                    CHAR(4) NOT NULL,
INVOICE_AMOUNT                 DECIMAL(7, 2),
INVOICE_DATE                   DATE,
FOREIGN KEY (CUSTOMER_ID) REFERENCES CUSTOMER)
```

Column name	Data type schema	Data type name	Column Length	Scale	Nulls
INVOICE_ID	SYSIBM	CHARACTER	7	0	No
CUSTOMER_ID	SYSIBM	CHARACTER	4	0	Yes
INVOICE_AMOUNT	SYSIBM	DECIMAL	7	2	Yes
INVOICE_DATE	SYSIBM	DATE	4	0	Yes

Credit Table

```
CREATE TABLE CREDIT (
CREDIT_ID              CHAR(7) NOT NULL PRIMARY KEY,
CUSTOMER_ID            CHAR(4) NOT NULL,
CUSTOMER_NAME          VARCHAR(20),
CREDIT_AMOUNT          DECIMAL (7,2),
FOREIGN KEY (CUSTOMER_ID) REFERENCES CUSTOMER,
FOREIGN KEY (CUSTOMER_NAME) REFERENCES CUSTOMER)
```

Column name	Data type schema	Data type name	Column Length	Scale	Nulls
CREDIT_ID	SYSIBM	CHARACTER	7	0	No
CUSTOMER_ID	SYSIBM	CHARACTER	4	0	No
CUSTOMER_NAME	SYSIBM	VARCHAR	20	0	Yes
CREDIT_AMOUNT	SYSIBM	DECIMAL	7	2	Yes



Rating Table

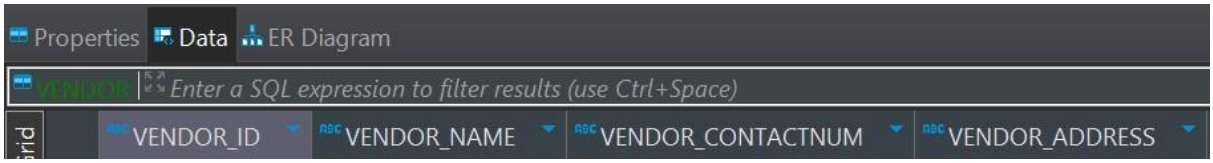
```
CREATE TABLE RATING (
RATING_ID              CHAR(7) NOT NULL PRIMARY KEY,
RATING_AMOUNT          INTEGER CHECK (RATING_AMOUNT>=0 AND
RATING_AMOUNT<=5),
PRODUCT_ID             CHAR(7) NOT NULL,
CUSTOMER_ID            CHAR(4) NOT NULL,
FOREIGN KEY (PRODUCT_ID) REFERENCES PRODUCT,
FOREIGN KEY (CUSTOMER_ID) REFERENCES CUSTOMER)
```

Column name	Data type schema	Data type name	Column Length	Scale	Nulls
RATING_ID	SYSIBM	CHARACTER	7	0	No
RATING_AMOUNT	SYSIBM	INTEGER	4	0	Yes
PRODUCT_ID	SYSIBM	CHARACTER	7	0	Yes
CUSTOMER_ID	SYSIBM	CHARACTER	4	0	Yes

Vendor Table

```
CREATE TABLE VENDOR (
VENDOR_ID           CHAR(7) NOT NULL PRIMARY KEY,
VENDOR_NAME         VARCHAR(20),
VENDOR_CONTACTNUM   VARCHAR(13),
VENDOR_ADDRESS      VARCHAR(40))
```

Column name	Data type schema	Data type name	Column Length	Scale	Nulls
VENDOR_ID	SYSIBM	CHARACTER	7	0	No
VENDOR_NAME	SYSIBM	VARCHAR	20	0	Yes
VENDOR_CONTACTNUM	SYSIBM	VARCHAR	13	0	Yes
VENDOR_ADDRESS	SYSIBM	VARCHAR	40	0	Yes



Voucher Table

```
CREATE TABLE VOUCHER (
VOUCHER_ID              CHAR(7) NOT NULL PRIMARY KEY,
VENDOR_ID               CHAR(7) NOT NULL,
CUSTOMER_ID             CHAR(4) NOT NULL,
DISCOUNT_PRICE          INT,
EXPIRY_DATE             DATE,
FOREIGN KEY (CUSTOMER_ID) REFERENCES CUSTOMER,
FOREIGN KEY (VENDOR_ID) REFERENCES VENDOR)
```

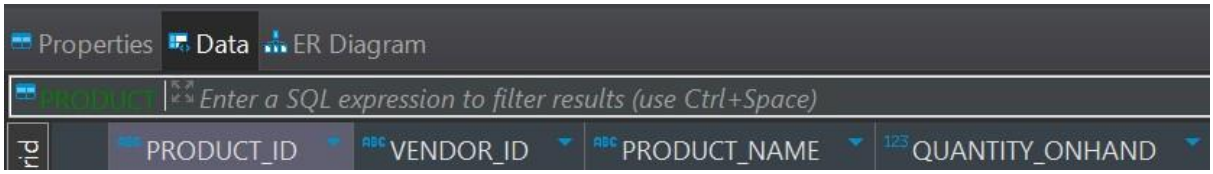
Column name	Data type schema	Data type name	Column Length	Scale	Nulls
VOUCHER_ID	SYSIBM	CHARACTER	7	0	No
VENDOR_ID	SYSIBM	CHARACTER	7	0	Yes
CUSTOMER_ID	SYSIBM	CHARACTER	4	0	Yes
DISCOUNT_PRICE	SYSIBM	INTEGER	4	0	Yes
EXPIRY_DATE	SYSIBM	DATE	4	0	Yes

Product Table

```
CREATE TABLE PRODUCT (
```

```
PRODUCT_ID          CHAR(7) NOT NULL PRIMARY KEY,
VENDOR_ID           CHAR(7) NOT NULL,
PRODUCT_NAME        VARCHAR(20),
QUANTITY_ONHAND         INT,
 FOREIGN KEY (VENDOR_ID) REFERENCES VENDOR)
```

Column name	Data type schema	Data type name	Column Length	Scale	Nulls
PRODUCT_ID	SYSIBM	CHARACTER	7	0	No
VENDOR_ID	SYSIBM	CHARACTER	7	0	No
PRODUCT_NAME	SYSIBM	VARCHAR	20	0	Yes
QUANTITY_ONHAND	SYSIBM	INTEGER	4	0	Yes



Listed Table

```
CREATE TABLE LISTED (
LISTING_ID          CHAR(7) NOT NULL PRIMARY KEY,
PRODUCT_ID          CHAR(7) NOT NULL,
CATEGORY_ID         CHAR(7) NOT NULL,
PRODUCT_NAME        VARCHAR(20),
PRICE               DECIMAL(7, 2),
FOREIGN KEY (PRODUCT_ID) REFERENCES PRODUCT,
FOREIGN KEY (CATEGORY_ID) REFERENCES CATEGORY,
FOREIGN KEY (PRODUCT_NAME) REFERENCES PRODUCT)
```

Column name	Data type schema	Data type name	Column Length	Scale	Nulls
LISTING_ID	SYSIBM	CHARACTER	7	0	No
PRODUCT_ID	SYSIBM	CHARACTER	7	0	No
CATEGORY_ID	SYSIBM	CHARACTER	7	0	No
PRODUCT_NAME	SYSIBM	VARCHAR	30	0	Yes
PRICE	SYSIBM	DECIMAL	7	2	Yes

Category Table

```
CREATE TABLE CATEGORY (
CATEGORY_ID             CHAR(7) NOT NULL PRIMARY KEY,
CATEGORY_NAME       VARCHAR(20),
CATEGORY_DESCRIPTION VARCHAR(30)
```

Column name	Data type schema	Data type name	Column Length	Scale	Nulls
CATEGORY_ID	SYSIBM	CHARACTER	7	0	No
CATEGORY_NAME	SYSIBM	VARCHAR	20	0	Yes
CATEGORY_DESCRIPTION	SYSIBM	VARCHAR	30	0	Yes

3. Data Insertion

Customer Table

INSERT INTO CUSTOMER(Customer_ID,Customer_Name, Customer_ContactNum,Customer_Address)
VALUES
(‘1001’, ‘Lily Peterson’, '60123456789', ‘123 Maple St, Springfield’),
(‘1002’, ‘Max Turner’, '60198765432', ‘456 Oak Ave, Meadowbrook’),
(‘1003’, ‘Ava Rodriguez’, '60162345678', ‘789 Pine Ln, Riverdale’),
(‘1004’, ‘Jake Mitchell’, '60189876543', ‘101 Elm Ct, Lakeside’),
(‘1005’, ‘Mia Chang’, '60178765432', ‘222 Cedar Rd, Hillcrest’),
(‘1006’, ‘Ethan Scott’, '60134567890', ‘333 Birch Dr, Sunset Valley’),
(‘1007’, ‘Emma White’, '60123456789', ‘444 Walnut Way, Greenfield’),
(‘1008’, ‘Alex Lee’, '60167890123', ‘555 Spruce Blvd, Willow Creek’),
(‘1009’, ‘Olivia Patel’, '60192345678', ‘666 Ash Lane, Pinecrest’), (‘1010’, ‘Noah Brown’, '60178901234', ‘777 Juniper Ave, Oakdale’)

	CUSTOMER_ID	CUSTOMER_NAME	CUSTOMER_CONTACTNUM	CUSTOMER_ADDRESS
1	1001	Lily Peterson	60123456789	123 Maple St, Springfield
2	1002	Max Turner	60198765432	456 Oak Ave, Meadowbrook
3	1003	Ava Rodriguez	60162345678	789 Pine Ln, Riverdale
4	1004	Jake Mitchell	60189876543	101 Elm Ct, Lakeside
5	1005	Mia Chang	60178765432	222 Cedar Rd, Hillcrest
6	1006	Ethan Scott	60134567890	333 Birch Dr, Sunset Valley
7	1007	Emma White	60123456789	444 Walnut Way, Greenfield
8	1008	Alex Lee	60167890123	555 Spruce Blvd, Willow Creek
9	1009	Olivia Patel	60192345678	666 Ash Lane, Pinecrest
10	1010	Noah Brown	60178901234	777 Juniper Ave, Oakdale

Order Table

INSERT INTO ORDER (ORDER_ID, CUSTOMER_ID, CUSTOMER_NAME, ORDER_DATE, QUANTITY)
VALUES
('ORD1001', '1000', 'Lily Peterson', '2024-02-15', 3),
('ORD1002', '1001', 'Max Turner', '2024-02-14', 5),
('ORD1003', '1002', 'Ava Rodriguez', '2024-02-13', 2),
('ORD1004', '1003', 'Jake Mitchell', '2024-02-12', 4),
('ORD1005', '1004', 'Mia Chang', '2024-02-11', 1),
('ORD1006', '1005', 'Ethan Scott', '2024-02-10', 6),
('ORD1007', '1006', 'Emma White', '2024-02-09', 3),
('ORD1008', '1007', 'Alex Lee', '2024-02-08', 2),
('ORD1009', '1008', 'Olivia Patel', '2024-02-07', 4),
('ORD1010', '1009', 'Noah Brown', '2024-02-06', 5)

	ORDER_ID	CUSTOMER_ID	CUSTOMER_NAME	ORDER_DATE	QUANTITY
1	ORD1001	1000	Lily Peterson	2024-02-15	3
2	ORD1002	1001	Max Turner	2024-02-14	5
3	ORD1003	1002	Ava Rodriguez	2024-02-13	2
4	ORD1004	1003	Jake Mitchell	2024-02-12	4
5	ORD1005	1004	Mia Chang	2024-02-11	1
6	ORD1006	1005	Ethan Scott	2024-02-10	6
7	ORD1007	1006	Emma White	2024-02-09	3
8	ORD1008	1007	Alex Lee	2024-02-08	2
9	ORD1009	1008	Olivia Patel	2024-02-07	4
10	ORD1010	1009	Noah Brown	2024-02-06	5

Tracking Table

INSERT INTO TRACKING (TRACKING_ID, ORDER_ID, STATUS, LOCATION)
VALUES
('TRC1001', 'ORD1001', 'Delivered', 'Springfield'),
('TRC1002', 'ORD1002', 'In Transit', 'Meadowbrook'),
('TRC1003', 'ORD1003', 'Out for Delivery', 'Riverdale'),
('TRC1004', 'ORD1004', 'Not Delivered', 'Lakeside'),
('TRC1005', 'ORD1005', 'Delivered', 'Hillcrest'),
('TRC1006', 'ORD1006', 'In Transit', 'Sunset Valley'),
('TRC1007', 'ORD1007', 'Out for Delivery', 'Greenfield'),
('TRC1008', 'ORD1008', 'Not Delivered', 'Willow Creek'),
('TRC1009', 'ORD1009', 'Delivered', 'Pinecrest'),
('TRC1010', 'ORD1010', 'In Transit', 'Oakdale');

	TRACKING_ID	ORDER_ID	STATUS	LOCATION
1	TRC1001	ORD1001	Delivered	Springfield
2	TRC1002	ORD1002	In Transit	Meadowbrook
3	TRC1003	ORD1003	Out for Delivery	Riverdale
4	TRC1004	ORD1004	Not Delivered	Lakeside
5	TRC1005	ORD1005	Delivered	Hillcrest
6	TRC1006	ORD1006	In Transit	Sunset Valley
7	TRC1007	ORD1007	Out for Delivery	Greenfield
8	TRC1008	ORD1008	Not Delivered	Willow Creek
9	TRC1009	ORD1009	Delivered	Pinecrest
10	TRC1010	ORD1010	In Transit	Oakdale

Invoice Table

INSERT INTO INVOICE (INVOICE_ID, CUSTOMER_ID, INVOICE_AMOUNT, INVOICE_DATE)
VALUES
('INV1001', '1000', 123.45, '2024-02-15'),
('INV1002', '1001', 234.56, '2024-02-14'),
('INV1003', '1002', 345.67, '2024-02-13'),
('INV1004', '1003', 456.78, '2024-02-12'),
('INV1005', '1004', 567.89, '2024-02-11'),
('INV1006', '1005', 678.90, '2024-02-10'),
('INV1007', '1006', 789.01, '2024-02-09'),
('INV1008', '1007', 890.12, '2024-02-08'),
('INV1009', '1008', 901.23, '2024-02-07'),
('INV1010', '1009', 1023.45, '2024-02-06');

	INVOICE_ID	CUSTOMER_ID	INVOICE_AMOUNT	INVOICE_DATE
1	INV1001	1000	123.45	2024-02-15
2	INV1002	1001	234.56	2024-02-14
3	INV1003	1002	345.67	2024-02-13
4	INV1004	1003	456.78	2024-02-12
5	INV1005	1004	567.89	2024-02-11
6	INV1006	1005	678.9	2024-02-10
7	INV1007	1006	789.01	2024-02-09
8	INV1008	1007	890.12	2024-02-08
9	INV1009	1008	901.23	2024-02-07
10	INV1010	1009	1,023.45	2024-02-06

Credit Table

INSERT INTO CREDIT (CREDIT_ID, CUSTOMER_ID, CUSTOMER_NAME, CREDIT_AMOUNT)
VALUES
('CRD1001', '1000',’Lily Peterson’, 50.00), ('CRD1002',
'1001', 'Max Turner', 30.00),
('CRD1003', '1002', 'Ava Rodriguez', 20.00),
('CRD1004', '1003', 'Jake Mitchell', 40.00),
('CRD1005', '1004', 'Mia Chang', 60.00),
('CRD1006', '1005', 'Ethan Scott', 70.00),
('CRD1007', '1006', 'Emma White', 80.00),
('CRD1008', '1007', 'Alex Lee', 90.00),
('CRD1009', '1008', 'Olivia Patel', 100.00),
('CRD1010', '1009', 'Noah Brown', 110.00);

Rating Table

INSERT INTO RATING (RATING_ID, RATING_AMOUNT, PRODUCT_ID, CUSTOMER_ID)
VALUES
('RTN1001', 4, 'PRD1001', '1000'),
('RTN1002', 3, 'PRD1002', '1001'),
('RTN1003', 5, 'PRD1003', '1002'),
('RTN1004', 2, 'PRD1004', '1003'),
('RTN1005', 1, 'PRD1005', '1004'),
('RTN1006', 4, 'PRD1006', '1005'),
('RTN1007', 3, 'PRD1007', '1006'),
('RTN1008', 5, 'PRD1008', '1007'),
('RTN1009', 2, 'PRD1009', '1008'),
('RTN1010', 1, 'PRD1010', '1009');

	RATING_ID	RATING_AMOUNT	PRODUCT_ID	CUSTOMER_ID
1	RTN1001	4	PRD1001	1000
2	RTN1002	3	PRD1002	1001
3	RTN1003	5	PRD1003	1002
4	RTN1004	2	PRD1004	1003
5	RTN1005	1	PRD1005	1004
6	RTN1006	4	PRD1006	1005
7	RTN1007	3	PRD1007	1006
8	RTN1008	5	PRD1008	1007
9	RTN1009	2	PRD1009	1008
10	RTN1010	1	PRD1010	1009

Vendor Table

INSERT INTO VENDOR (VENDOR_ID, VENDOR_NAME, VENDOR_CONTACTNUM, VENDOR_ADDRESS)
VALUES
('VND1001', 'ABC Company', '60123456789', '123 Main St, Cityville'),
('VND1002', 'XYZ Corporation', '60198765432', '456 Elm St, Townsville'),
('VND1003', 'LMN Enterprises', '60162345678', '789 Oak St, Village Town'),
('VND1004', 'PQR Ltd.', '60189876543', '101 Maple St, Hamletville'),
('VND1005', 'UVW Inc.', '60178765432', '222 Pine St, Countryside');

	VENDOR_ID	VENDOR_NAME	VENDOR_CONTACTNUM	VENDOR_ADDRESS
1	VND1001	ABC Company	60123456789	123 Main St, Cityville
2	VND1002	XYZ Corporation	60198765432	456 Elm St, Townsville
3	VND1003	LMN Enterprises	60162345678	789 Oak St, Village Town
4	VND1004	PQR Ltd.	60189876543	101 Maple St, Hamletville
5	VND1005	UVW Inc.	60178765432	222 Pine St, Countryside

Voucher Table

INSERT INTO VOUCHER (VOUCHER_ID, VENDOR_ID, CUSTOMER_ID, DISCOUNT_PRICE, EXPIRY_DATE)
VALUES
('VCH1001', 'VND1001', '1000', 10, '2024-03-31'),
('VCH1002', 'VND1002', '1001', 15, '2024-04-30'),
('VCH1003', 'VND1003', '1002', 20, '2024-05-31'),
('VCH1004', 'VND1004', '1003', 25, '2024-06-30'),
('VCH1005', 'VND1005', '1004', 30, '2024-07-31');

	VOUCHER_ID	VENDOR_ID	CUSTOMER_ID	DISCOUNT_PRICE	EXPIRY_DATE
1	VCH1001	VND1001	1000	10	2024-03-31
2	VCH1002	VND1002	1001	15	2024-04-30
3	VCH1003	VND1003	1002	20	2024-05-31
4	VCH1004	VND1004	1003	25	2024-06-30
5	VCH1005	VND1005	1004	30	2024-07-31

Product Table

INSERT INTO PRODUCT (PRODUCT_ID, VENDOR_ID, PRODUCT_NAME, QUANTITY_ONHAND)
VALUES
('PRD1001', 'VND1001', 'Dell XPS Laptop', 50),
('PRD1002', 'VND1002', 'iPhone 13 Pro', 100),
('PRD1003', 'VND1003', 'Harry Potter', 75),
('PRD1004', 'VND1004', 'Nike Football', 30), ('PRD1005',
'VND1005', 'Ikea Double bed', 80);

	PRODUCT_ID	VENDOR_ID	PRODUCT_NAME	QUANTITY_ONHAND
1	PRD1001	VND1001	Dell xps Laptop	50
2	PRD1002	VND1002	iPhone 13 Pro	100
3	PRD1003	VND1003	Harry Potter	75
4	PRD1004	VND1004	Nike Football	30
5	PRD1005	VND1005	Ikea Double bed	80

Listed Table

INSERT INTO LISTED (LISTING_ID, PRODUCT_ID, CATEGORY_ID, PRODUCT_NAME, PRICE)
VALUES
('LST1001', 'PRD1001', 'CAT1001', 'Dell XPS Laptop', 799.99),
('LST1002', 'PRD1002', 'CAT1001', 'iPhone 13 Pro', 999.99),
('LST1003', 'PRD1003', 'CAT1001', 'Harry Potter', 699.99),
('LST1004', 'PRD1004', 'CAT1002', 'Nike football', 1999.99),
('LST1005', 'PRD1005', 'CAT1003', 'Ikea Double bed', 349.99)

	LISTING_ID	PRODUCT_ID	CATEGORY_ID	PRODUCT_NAME	PRICE
1	LST1001	PRD1001	CAT1001	Dell xps Laptop	799.99
2	LST1002	PRD1002	CAT1001	iPhone 13 Pro	999.99
3	LST1003	PRD1003	CAT1003	Harry Potter	699.99
4	LST1004	PRD1004	CAT1004	Nike football	1,999.99
5	LST1005	PRD1005	CAT1005	Ikea Double bed	349.99

Category Table

INSERT INTO CATEGORY (CATEGORY_ID, CATEGORY_NAME, CATEGORY_DESCRIPTION)
VALUES
('CAT1001', 'Electronics', 'Devices and gadgets'),
('CAT1002', 'Clothing', 'Apparel and fashion items'),
('CAT1003', 'Books', 'Books and literature'), ('CAT1004',
‘Fitness', 'Exercise equipment'),
('CAT1005', 'Furniture', 'Furniture and Home decorations');

	CATEGORY_ID	CATEGORY_NAME	CATEGORY_DESCRIPTION
1	CAT1001	Electronics	Devices and gadgets
2	CAT1002	Clothing	Apparel and fashion items
3	CAT1003	Books	Books and literature
4	CAT1004	Fitness	Exercise equipment
5	CAT1005	Furniture	Furniture and Home decorations

4. Data Manipulation using SQL

I. Aggregate Function

An Aggregate function is a function that operates on a set of values and returns a single aggregated value summarising those values. For example, the average of all values, the sum of all the values and the maximum or minimum of a certain group of values. In the program, we used **all five aggregate functions** which are maximum, minimum, count, sum and average.

1. Max

- In Figure 1, the max function is used to select the product with the highest price which is **1999.99**

SELECT MAX(PRICE) FROM LISTED

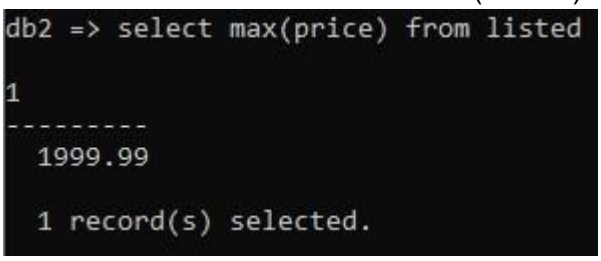


Figure 1

2. Min

- In Figure 2, the min function is used to select the product with the smallest price which is **349.99**

SELECT MIN(PRICE) FROM LISTED

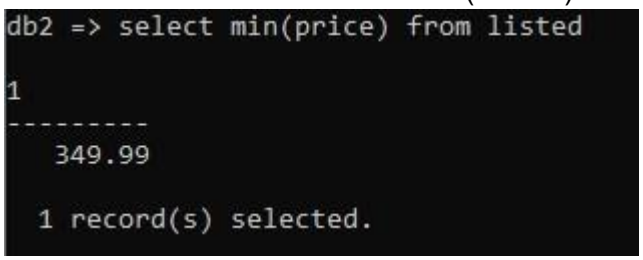


Figure 2

3. Count

- In Figure 3, the count function is used to count the number of products which were marked delivered which is **3**

SELECT COUNT FROM TRACKING WHERE STATUS = ‘Delivered’

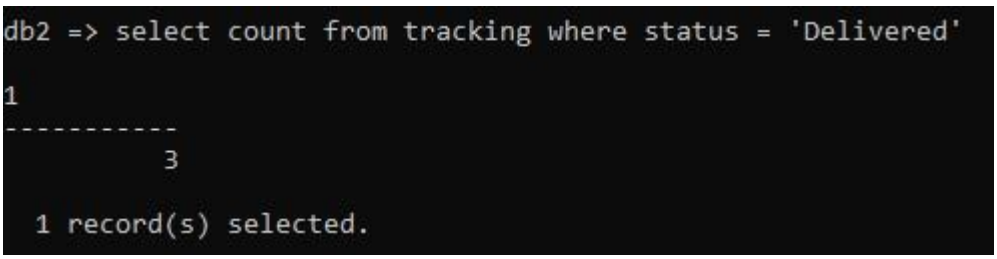


Figure 3

4. Sum

- In Figure 4, the sum function is used to calculate the sum of price of the products listed which is **4849.95**

SELECT SUM(PRICE) FROM LISTED

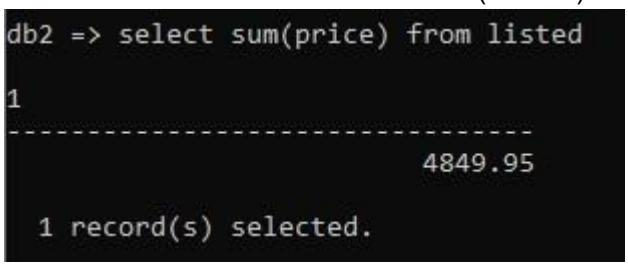


Figure 4

5. Avg

- In Figure 5, the avg function is used to calculate the average of price of the products listed which is **969.99**

SELECT AVG(PRICE) FROM LISTED

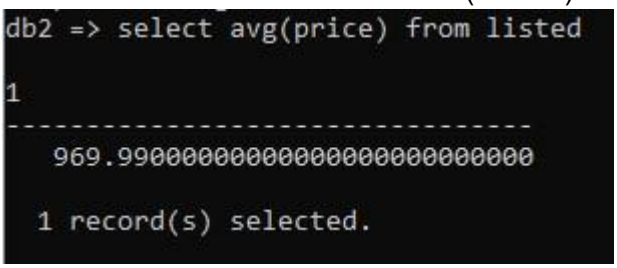
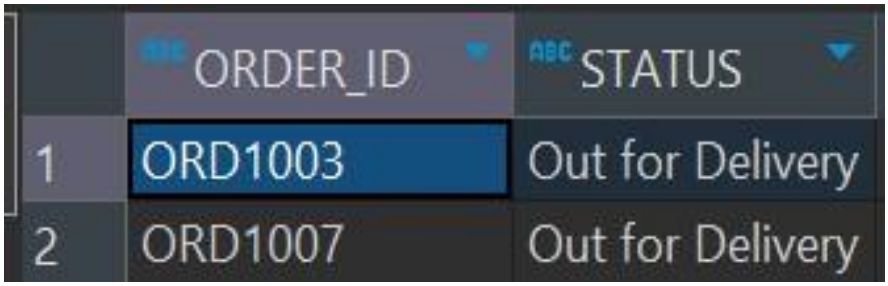


Figure 5

II. View

This view displays the orders that are out for delivery, so when a customer calls and tells their order id to check if it's out for delivery we can check this view.

```
CREATE VIEW ORDERS_OUT_FOR_DELIVERY AS
SELECT ORDER_ID, STATUS
FROM TRACKING
WHERE STATUS = 'Out for Delivery';
```



	ORDER_ID	STATUS
1	ORD1003	Out for Delivery
2	ORD1007	Out for Delivery

III. One subquery/nested query

A nested query that finds the customers who have placed the most orders in each location. This could be useful for a sales manager to identify potential high-value customers in each location.

```sql
SELECT
    T.LOCATION AS "Location",
    O.CUSTOMER_NAME AS "Customer Name",
    COUNT(O.ORDER_ID) AS "Number of Orders"
FROM
    "ORDER" O
JOIN
    TRACKING T ON O.ORDER_ID = T.ORDER_ID
GROUP BY
    T.LOCATION,
    O.CUSTOMER_NAME
HAVING
    COUNT(O.ORDER_ID) = (
        SELECT
            MAX(OrderCount)
        FROM (
            SELECT
                LOCATION,
                COUNT(O.ORDER_ID) AS OrderCount
            FROM
                "ORDER" O
            JOIN
                TRACKING T ON O.ORDER_ID = T.ORDER_ID
            GROUP BY
                LOCATION
        ) AS SubQuery
        WHERE
            SubQuery.LOCATION = T.LOCATION
    );
```

For data inserted in this database:

	Location	Customer Name	Number of Orders
1	Greenfield	Emma White	1
2	Hillcrest	Mia Chang	1
3	Lakeside	Jake Mitchell	1
4	Meadowbrook	Max Turner	1
5	Oakdale	Noah Brown	1
6	Pinecrest	Olivia Patel	1
7	Riverdale	Ava Rodriguez	1
8	Springfield	Lily Peterson	1
9	Sunset Valley	Ethan Scott	1
10	Willow Creek	Alex Lee	1

IV. One query with a 'group by' and 'having' clauses

This query finds the total product in each category(if there are products listed in that category).

```
SELECT CATEGORY_ID, COUNT(*) AS total_products
FROM LISTED
```

```
GROUP BY CATEGORY_ID
HAVING COUNT(*) < 5;
```

	CATEGORY_ID	TOTAL_PRODUCTS
1	CAT1001	2
2	CAT1003	1
3	CAT1004	1
4	CAT1005	1

V. Trigger

This is used to update product quantity on hand when a product is listed.

```
CREATE TRIGGER update_quantity_onhand_after_listing1
AFTER
INSERT
    ON
    LISTED
REFERENCING NEW AS N
FOR EACH ROW
MODE DB2SQL
UPDATE
    PRODUCT
SET
    QUANTITY_ONHAND = QUANTITY_ONHAND - 1
WHERE
    PRODUCT_NAME = N.PRODUCT_NAME
```

In this example Dell xps Laptop is listed on record 6, and quantity on hand is updated from 50 to 49 in the PRODUCT table.

Before listed:

	LISTING_ID	PRODUCT_ID	CATEGORY_ID	PRODUCT_NAME	PRICE
1	LST1001	PRD1001	CAT1001	Dell xps Laptop	799.99
2	LST1002	PRD1002	CAT1001	iPhone 13 Pro	999.99
3	LST1003	PRD1003	CAT1003	Harry Potter	699.99
4	LST1004	PRD1004	CAT1004	Nike football	1,999.99
5	LST1005	PRD1005	CAT1005	Ikea Double bed	349.99

Add PRD1001:

	LISTING_ID	PRODUCT_ID	CATEGORY_ID	PRODUCT_NAME	PRICE
1	LST1001	PRD1001	CAT1001	Dell xps Laptop	799.99
2	LST1002	PRD1002	CAT1001	iPhone 13 Pro	999.99
3	LST1003	PRD1003	CAT1003	Harry Potter	699.99
4	LST1004	PRD1004	CAT1004	Nike football	1,999.99
5	LST1005	PRD1005	CAT1005	Ikea Double bed	349.99
6	LST1006	PRD1001	CAT1001	Dell xps Laptop	799.99

Before PRODUCT Table:

	PRODUCT_ID	VENDOR_ID	PRODUCT_NAME	QUANTITY_ONHAND
1	PRD1001	VND1001	Dell xps Laptop	50
2	PRD1002	VND1002	iPhone 13 Pro	100
3	PRD1003	VND1003	Harry Potter	75
4	PRD1004	VND1004	Nike Football	30
5	PRD1005	VND1005	Ikea Double bed	80

After PRODUCT Table:

	PRODUCT_ID	VENDOR_ID	PRODUCT_NAME	QUANTITY_ONHAND
1	PRD1001	VND1001	Dell xps Laptop	49
2	PRD1002	VND1002	iPhone 13 Pro	100
3	PRD1003	VND1003	Harry Potter	75
4	PRD1004	VND1004	Nike Football	30
5	PRD1005	VND1005	Ikea Double bed	80

VI. Stored procedures

```
CREATE PROCEDURE GetVendorProducts (IN VendorId VARCHAR(50))
LANGUAGE SQL
BEGIN
    DECLARE @sql VARCHAR(500);
    SET @sql = 'SELECT P.PRODUCT_ID, P.PRODUCT_NAME, P.QUANTITY_ONHAND
    FROM PRODUCT P
    WHERE P.VENDOR_ID = ?';
    PREPARE stmt FROM @sql;
    EXECUTE stmt USING VendorId;
END
```

Calling example: CALL GetVendorProducts('VND1001');

VII. Four Queries Not Covered in Lecture

1. Rand()

The RAND() function is used to generate a random number in SQL. For this usage, we can pick a random product for users to try out when they are indecisive.

```
SELECT PRODUCT_ID, VENDOR_ID, PRODUCT_NAME
FROM PRODUCT
ORDER BY RAND()
FETCH FIRST 3 ROWS ONLY
```

```
db2 => select product_id, vendor_id, product_name from product order by rand() fetch first 3 rows only

PRODUCT_ID VENDOR_ID PRODUCT_NAME
---------- --------- --------------------
PRD1003    VND1003   Harry Potter
PRD1001    VND1001   Dell XPS Laptop
PRD1005    VND1005   Ikea Double bed

  3 record(s) selected.
```

2. Using Fetch

For example, in large databases using the SELECT statement to query data from a table will get you a large number of rows. By using the FETCH query, we can limit the number of rows that will be returned by the query.

The result will show us the top 5 products based on the rating when being run on our database.

```
SELECT R.CUSTOMER_ID, C.CUSTOMER_NAME, R.RATING_AMOUNT,
R.PRODUCT_ID FROM RATING R
JOIN CUSTOMER C ON R.CUSTOMER_ID = C.CUSTOMER_ID
ORDER BY R.RATING_AMOUNT DESC
FETCH FIRST 5 ROWS ONLY
```

```
db2 => select r.customer_id, c.customer_name, r.rating_amount, r.product_id from rating r join customer c on r.customer_id =
c.customer_id order by r.rating_amount desc fetch first 5 rows only

CUSTOMER_ID CUSTOMER_NAME        RATING_AMOUNT PRODUCT_ID
----------- -------------------- ------------- ----------
1002        Ava Rodriguez                    5 PRD1003
1007        Alex Lee                         5 PRD1008
1000        Lily Peterson                    4 PRD1001
1005        Ethan Scott                      4 PRD1006
1001        Max Turner                       3 PRD1002

  5 record(s) selected.
```

3. Ranking Query

By using the ranking query, we can request that DB2 rank for us the price of the products from lowest price to the highest and sort them by ranks

```
SELECT PRODUCT_ID, PRODUCT_NAME, PRICE
RANK() OVER (ORDER BY PRICE)
AS RANK
FROM LISTED
ORDER BY RANK
```

```
db2 => SELECT PRODUCT_ID, PRODUCT_NAME, PRICE, RANK() OVER (ORDER BY PRICE) AS rank FROM LISTED ORDER BY rank

PRODUCT_ID PRODUCT_NAME         PRICE     rank
---------- -------------------- --------- --------------------
PRD1005    Ikea Double bed         349.99                    1
PRD1003    Harry Potter            699.99                    2
PRD1001    Dell XPS Laptop         799.99                    3
PRD1002    iPhone 13 Pro           999.99                    4
PRD1004    Nike Football          1999.99                    5

  5 record(s) selected.
```