1. Project Description

The project consists in the creation of a database required to organize a company that it is developing its business in the IT&C market. The company sales hardware products and offers support services for the hardware products of its customers (the same products that can be found in their own portfolio). The company employs sales persons which are to be assigned specific customers. Each sales person serves its specific customers for which they handle orders or support contracts. Prior to booking a contract or making an order customers may also requests quotes to check the prices. Customers can receive different discounts based on the length of the support contract they are about to book (discounts are offered only for support contracts). The company obtains their products from different suppliers, having a different supplier for each product line. The company also employees engineers, specialized or not on different products, that can provide onsite support services for their customers.

2. Identification of Business Rules

1. A customer has only 1 sales person assigned to handle its account.
2. For support contracts with the length of 2 years a discount of 3% is applied, for duration of 3 years a 4% discount is applied, for 4 years a 5% discount, and for a length of more than 5 years a 7% discount is applied.
3. Each supplier provides a single product line (type of product) for the company.
4. It is mandatory for a customer to have a correct VAT ID (for financial rules).
5. The onsite support does not have a fixed price; it is calculated based on the time spent by the engineer onsite, customer's location and the type of problem.
6. Engineers don't have to be specialized in a product line.
7. Quotes are only provided for support contracts.
8. The company cannot provide support services for the same customer at two different locations.
9. A quote with a start date in the past cannot be activated.
10. A quote is available only for 30 days after its creation.
11. Sale orders (type 3 n Order_call table) don't need a start date/ end date.

3. Identification of Assumptions Made:

1. A customer can have more than one Contact persons (ordering contact) – the contact person is specified in the Order-Call table (for example a customer can have many departments and can have for each department a different order; different contact person).
2. The VAT ID standard format is IE + 7 digits and 2 letters. Ex: IE1234567FA.

3. The support price is assumed to be for 8 hours a day, 5 days a week support services, which is the single support level agreement provided by the company; in tables, the monthly support price is used.
4. Support contract is offered only for products that company offers in their product catalog (are available for sale in the Product table).
5. Sales persons can have more than 1 customer to look after.
6. I have assumed that an onsite support from a customer can be requested by any employee from the customer, and a contact person entity is not needed in this case.
7. Assuming the item price of a product will change over time, I am storing it in the order line table, so that past order values can be retrieved if required (if the item price is not stored, a past order value will retrieve the new price, even if the prices have been changed in the meantime).

4. Identification of entities, relationships, cardinality and optionality:

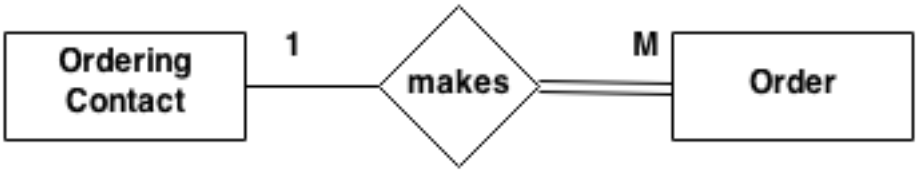
Entities

- Sales person
- Customer
- Contact Person
- Onsite support
- Engineer
- Product
- Supplier
- Order
- Order Line

- **Diagrams of individual relationships (2 entities and 1 relationship per diagram) in addition to a single line of explanation. Redundant relationships appear here but are crossed out in red.**

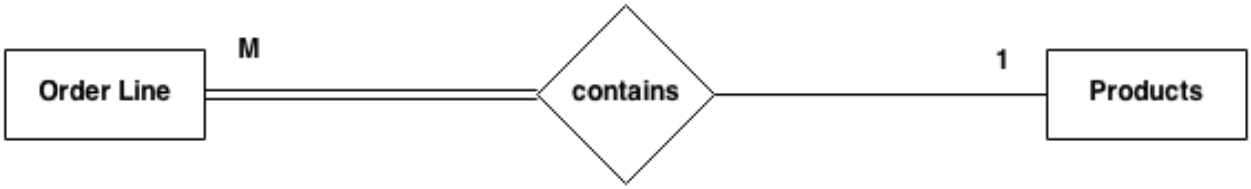
- One sales person is assigned many customers, and many customers are assigned to one sales person; a sales person can exist without having a customer assigned, but there is no customer without a sales person assigned, this being an optional to mandatory relationship.

- One customer can have many ordering contacts, but one ordering contact is assigned to only one customer; one ordering contact cannot exist without a customer, but a customer can exist without having an ordering contact (it didn't make any order request); this is an optional to mandatory relationship.

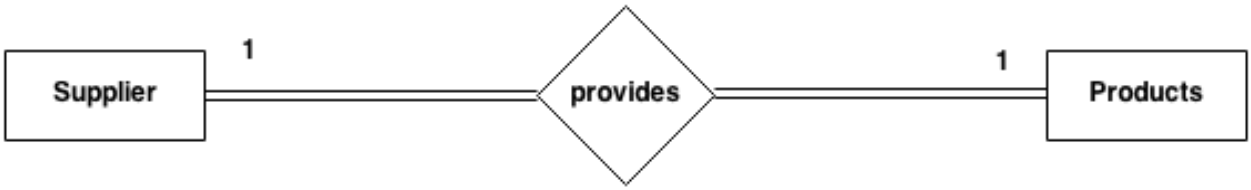


- One ordering contact can make many orders, but one order can have only one ordering contact; one ordering contact can be registered in the database without having an order, but an order cannot be registered without an ordering contact.

- One order can have many order lines, while an order line is linked to only one order; the order cannot exist without the order line and the order line cannot exist without the order – this is a mandatory to mandatory relationship.

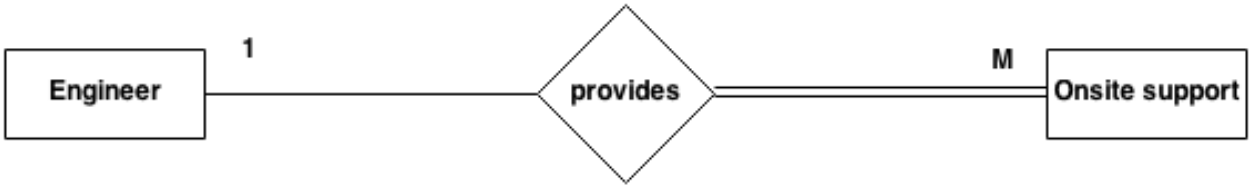


- One product is present on many order lines, while many order lines have only one product; an order line must have a product, while a product can exist without having an order line – this is a mandatory to optional relationship.

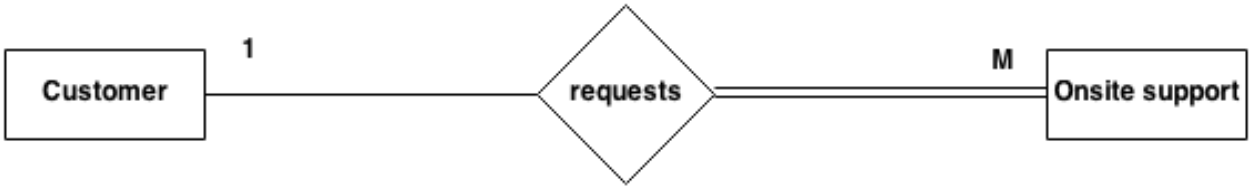


- One supplier provides only one product line, while one product line is provided by only one supplier; it is mandatory for a supplier to provide a product line, and it is mandatory for a product line to have a supplier – this is a mandatory to mandatory relationship.

- Many engineers are specialized in many different products and many products have many engineers with a specialization; an engineer might exist without having a specialization in a product line but having a general specialization, while product can exist as well without having an engineer specialized in the product line – this is an optional to optional relationship.

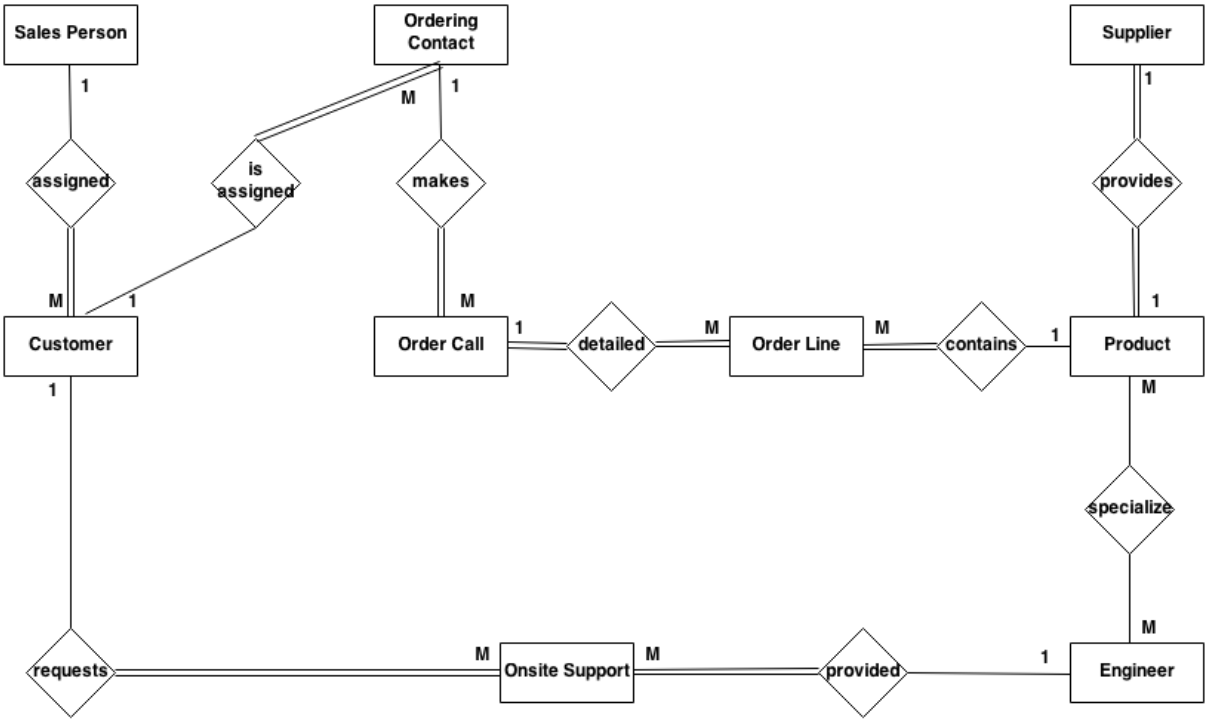


- One engineer can provide many onsite support services, but an onsite support service is delivered by one engineer; an engineer can exist without having an onsite support, but an onsite support service cannot exist without an engineer – this is an optional to mandatory relationship.



- One customer can have many onsite supports, but an onsite support can have only one customer; a customer can exist without having an onsite support, but an onsite support must have a customer, this being an optional to mandatory relationship.

5. Develop a prototype ERD for your project.



Part 2

A. Tables and Attributes Diagram underlining Primary Keys

1. Sales person (salesID, Fname, Lname, tel_no, emailID, address)
2. Customer (customerID, address, VATid, *salesID,* cname)
3. Onsite support (supportId, *customerID*, *engID*, time, duration, tcost)
4. Engineer (engID, fname, lname, tel_no, emailID, address, *spec*)
5. Supplier (supplierID, sname, address, scontname, tel_no, emailID)
6. Product (part_no, *supplier,* description, sale_price, sup_price, qty)
7. Order_call (orderID, order_date, type, start_date, end_date, , discount, *contactID*)
8. Order_line (orderID, part_no, qty, item_price)
9. Contact_person (contactID, *customerID,* cname, tel_no, emailID)

SALES_PERSON	CUSTOMER	ONSITE_SUPPORT	ENGINEER	SUPPLIER
salesID	customerID	supportID	engID	supplierID
Fname	address	*customerID*	fname	sname
Lname	VATID	*engID*	lname	address
tel_no	*salesID*	time	tel_no	scontname
emailID	cname	duration	emailID	tel_no
address		tcost	address	emailID
			spec	

PRODUCT	ORDER_CALL	ORDER_LINE	CONTACT_PERSON
part_no	orderID	orderID	contactID
supplier	order_date	part_no	*customerID*
description	type	qty	cname
sale_price	start_date	item_price	tel_no
sup_price	end_date		emailID
qty	discount		
	contactID		

B. Changes since the original Final Project Specification Submission.

The following changes have been made from the previous version of the project specifications:

- A new entity has been added: **CONTACT PERSON** – this entity refers to the person that makes an order call (from customer's side) and can be contacted related to the order calls he have made; this entity has split the relationship between the customer and the order;
- The relationships involving this new entities have been described;
- The Quote, Quote Line, Support Contract and Contract Line entities have been removed and all the information that could have been stored in those entities it will be stored in the Order Call and Order Line entities;
- The Business Rules section has been updated;
- The Assumptions sections has been updated;

1. Database Setup

```
-- Sales_person table creation
CREATE TABLE SALES_PERSON(
salesID NUMBER(10) NOT NULL,
Fname NVARCHAR2(30),
Lname NVARCHAR2(30),
tel_no VARCHAR(15),
emailID NVARCHAR2(30),
address NVARCHAR2(50),
CONSTRAINT SALES_PERSON_PK PRIMARY KEY (salesID));
```

Column Name	Data Type	Nullable	Default	Primary Key
SALESID	NUMBER(10,0)	No	-	1
FNAME	NVARCHAR2(30)	Yes	-	-
LNAME	NVARCHAR2(30)	Yes	-	-
TEL_NO	VARCHAR2(15)	Yes	-	-
EMAILID	NVARCHAR2(30)	Yes	-	-
ADDRESS	NVARCHAR2(50)	Yes	-	-

1 - 6

Data insertion

EDIT	SALESID	FNAME	LNAME	TEL_NO	EMAILID	ADDRESS
	100	Ollie	Suarez	0871234332	ollie_suarez@mycomp.com	32 Belfield, Dublin 4, Ireland
	101	Sandra	Marie	0871234554	sandra_marie@mycomp.com	36 Stillorgan, Dublin 4, Ireland
	102	Eoin	McBeth	0871190552	eoin_mcbeth@mycomp.com	14 Donnybrook, Dublin 4, Ireland
	103	Andrew	Slaymaker	0831494554	andrew_slaymaker@mycomp.com	36 Emmet Road, Dublin 8, Ireland
	104	Joan	Hodge	0891494674	joan_hodge@mycomp.com	36 Oconnel st, Dublin 2, Ireland
	105	Martin	Oneill	0821234678	martin_oneill@mycomp.com	36 Pembroke Rd, Dublin 3, Ireland
	106	Henry	Cavil	0831234567	henry_cavil@mycomp.com	5 Baggott st, Dublin 3, Ireland
	107	Maria	Heart	0821235438	maria_heart@mycomp.com	8 Grafton st, Dublin 2, Ireland
	108	Jon	Gil	0854367812	jon_gil@mycomp.com	17 Essex st, Dublin 5, Ireland
	109	Frank	Ohara	0854567832	frank_ohara@mycomp.com	56 Sussex st, Dublin 7, Ireland
	110	Jill	Hart	0834356217	jill_hart@mycomp.com	23 Avon st, Dublin 11, Ireland

row(s) 1 - 11 of 11

```
-- Customer table creation
CREATE TABLE CUSTOMER(
customerID NUMBER(15) NOT NULL,
caddress NVARCHAR2(50),
vatID NVARCHAR2(10) NOT NULL,
salesID NUMBER(10),
cname NVARCHAR2(30),
CONSTRAINT CUSTOMER_PK PRIMARY KEY (customerID),
CONSTRAINT CUSTOMER_FK FOREIGN KEY (salesID) REFERENCES SALES_PERSON(salesID));
```

Column Name	Data Type	Nullable	Default	Primary Key
CUSTOMERID	NUMBER(15,0)	No	-	1
CADDRESS	NVARCHAR2(50)	Yes	-	-
VATID	NVARCHAR2(10)	No	-	-
SALESID	NUMBER(10,0)	Yes	-	-
CNAME	NVARCHAR2(30)	Yes	-	-

1 - 5

Data insertion

EDIT	CUSTOMERID	CADDRESS	VATID	SALESID	CNAME
	506	81 Sandyford, Dublin 11, Ireland	IE9876543DA	102	Eriksen ltd
	507	9 Sandyford, Dublin 11, Ireland	IE1234567GA	101	Michelin
	508	8 Pembroke road, Dublin 3, Ireland	IE8769503FE	108	Hewlett Packard
	509	8 Stillorgan Road, Dublin 4, Ireland	IE4673219CA	108	UCD
	510	8 Pembroke road, Dublin 3, Ireland	IE4563165TR	101	Google
	511	4 Belfield, Dublin 4, Ireland	IE3451467FE	107	Marc Jacobs
	512	12 Lansdowne, Dublin 2, Ireland	IE5671234FE	106	IPA
	513	15 Emmet Road, Dublin 8, Ireland	IE1234791WE	106	Technologies
	514	1 Oconnel Street, Dublin 2, Ireland	IE5128903MC	103	Heineken
	515	10 Lansdowne, Dublin 3, Ireland	IE5419053DF	106	IRFU
	516	10 Lansdowne, Dublin 3, Ireland	IE1269012DF	109	Laserprint
	517	12 Tallaght road, Dublin 21, Ireland	IE1239041DF	110	Masters
	518	18 Essex st, Dublin 18, Ireland	IE5431784PO	109	Adidas
	519	15 Reading road, Dublin 19, Ireland	IE1290321DF	106	Oxfam
	520	7 George St, Dublin 24, Ireland	IE1238934IL	105	Jameson

row(s) 1 - 15 of 16

```
-- Contact person table creation
CREATE TABLE CONTACT_PERSON(
contactID NUMBER(15) NOT NULL,
customerID NUMBER(15) NOT NULL,
cname NVARCHAR2(50),
tel_no VARCHAR(15),
emailID NVARCHAR2(30),
CONSTRAINT CONTACT_PERSON_PK PRIMARY KEY (contactID),
CONSTRAINT CONTACT_PERSON_FK FOREIGN KEY (customerID) REFERENCES
CUSTOMER(customerID));
```

Column Name	Data Type	Nullable	Default	Primary Key
CONTACTID	NUMBER(15,0)	No	-	1
CUSTOMERID	NUMBER(15,0)	No	-	-
CNAME	NVARCHAR2(50)	Yes	-	-
TEL_NO	VARCHAR2(15)	Yes	-	-
EMAILID	NVARCHAR2(30)	Yes	-	-

1 - 5

Data insertion

EDIT	CONTACTID	CUSTOMERID	CNAME	TEL_NO	EMAILID
	1500	506	Alan Rogers	0981232149	alan_rogers@erikssen.ie
	1501	507	Fred Astaire	0832123567	fred.astaire@michelin.ie
	1502	508	Marc Ohara	0865431789	marc_ohara@hp.com
	1503	509	Frank Obama	0831256432	f_obama@ucd.ie
	1504	510	Ruby Tuesday	0831245321	r.tuesday@google.ie
	1505	511	Alan Pardew	0831256123	a_pardew@jacobs.ie
	1506	512	Louise Her	0843456765	l_her@ipa.ie
	1507	513	Mary Poppins	0831256123	m_poppins@tech.ie
	1508	514	Mark James	0831123213	mjames@heineken.ie
	1509	515	Grant Mar	0831123678	gmar@irfu.ie
	1510	516	Niamh Brothar	0835654578	n.brothar@laser.ie
	1511	517	James Joyce	0831123888	jjoyce@masters.ie
	1512	518	Neil Borg	0878787654	neil_borg@adidas.ie
	1513	519	Ozzie Osbourne	0831178787	ozzie@oxfam.ie
	1514	511	Rob Power	0838989898	rob_power@jacobs.ie

row(s) 1 - 15 of 16

```
-- SUPPLIER table creation
CREATE TABLE SUPPLIER(
supplierID NUMBER(10) NOT NULL,
sname NVARCHAR2(30),
address NVARCHAR2(50),
scontname NVARCHAR2(30),
tel_no VARCHAR(15),
emailID NVARCHAR2(30),
CONSTRAINT SUPPLIER_PK PRIMARY KEY (supplierID));
```

Column Name	Data Type	Nullable	Default	Primary Key
SUPPLIERID	NUMBER(10,0)	No	-	1
SNAME	NVARCHAR2(30)	Yes	-	-
ADDRESS	NVARCHAR2(50)	Yes	-	-
SCONTNAME	NVARCHAR2(30)	Yes	-	-
TEL_NO	VARCHAR2(15)	Yes	-	-
EMAILID	NVARCHAR2(30)	Yes	-	-

1 - 6

Data insertion

EDIT	SUPPLIERID	SNAME	ADDRESS	SCONTNAME	TEL_NO	EMAILID
	3020	Hewlett Packard	18 Stillorgan, Dublin 4	Alan Smith	0871234321	alansmith@hp.com
	3021	Sony	25 Sandyford, Dublin 11	Frank Jones	0871123989	h_jones@sony.com
	3022	IBM	10 Baggot Street, Dublin 3	Orla Jackson	0871545456	orla.jackson@ibm.com
	3023	HCL Technologies	18 Blanchardstown, Dublin 9	Steve Mark	0871243219	steve_mark@hcl.com
	3024	Red Hat	31 Essesx Street, Wexford	Joan Cussack	0813456543	j.cusack@redhat.com
	3025	Oracle	23 George Street, Dublin 21	Teddy Smith	0821232789	teddy.smith@oracle.com
	3026	Ericsson	24 Stillorgan, Dublin 4	Michaela Pearson	0812323470	mpearson@ericsson.com
	3027	Samsung Electronics	Stillorgan, Blackrock, Dublin 4	Steve Oneill	0874343432	soneill@samsung.com
	3028	Phillips	County Business Park, Dublin 18	Mark Grady	0813245765	mark.gready@phillips.com
	3029	Apple	Hollyhill Industrial Estate, Cork	Anne Fitzgerald	0856473095	anne_fitzgerald@apple.com
	3030	Nokia	Block 1B Beech Hill Office Campus Dublin Dublin 4	Miriam Montgomerry	0823234567	miriam.m@nokia.com

row(s) 1 - 11 of 11

```
-- Product table creation
CREATE TABLE PRODUCT(
part_no NUMBER(10) NOT NULL,
supplier NUMBER(10),
description NVARCHAR2(30),
sale_price NUMBER(10, 2),
sup_price NUMBER(5, 2),
qty NUMBER(5),
CONSTRAINT PRODUCT_PK PRIMARY KEY (part_no),
CONSTRAINT PRODUCT_FK FOREIGN KEY (supplier) REFERENCES SUPPLIER(supplierID));
```

Column Name	Data Type	Nullable	Default	Primary Key
PART_NO	NUMBER(10,0)	No	-	1
SUPPLIER	NUMBER(10,0)	Yes	-	-
DESCRIPTION	NVARCHAR2(30)	Yes	-	-
SALE_PRICE	NUMBER(10,2)	Yes	-	-
SUP_PRICE	NUMBER(5,2)	Yes	-	-
QTY	NUMBER(5,0)	Yes	-	-

1 - 6

Data insertion

EDIT	PART_NO	SUPPLIER	DESCRIPTION	SALE_PRICE	SUP_PRICE	QTY
	101	3026	Ericsson B and R Mgm	1000	50.5	20
	125	3025	Oracle Acme Packet 3820	650	35.75	60
	175	3030	Nokia Lumia 1020	559	15.85	30
	188	3021	Sony X25 Scanner	420	45	50
	190	3024	Red Hat Storage Server	800	65	20
	255	3028	Phillips X255 Printer	650	40.25	40
	295	3023	HP DL320 Server	1800	75	40
	320	3022	IBM 6182 Plotter	1250	80	35
	520	3029	Apple MacPro Dual GPU	3990	125	30
	521	3020	HP DL380p Gen8 Svr	2800	45.95	30
	860	3027	Samsung LT Cluster	2450	85.99	40

row(s) 1 - 11 of 11

```
-- ENGINEER table creation
CREATE TABLE ENGINEER(
engID NUMBER(10) NOT NULL,
Fname NVARCHAR2(30),
Lname NVARCHAR2(30),
tel_no VARCHAR(15),
emailID NVARCHAR2(30),
address NVARCHAR2(50),
spec NUMBER(10),
CONSTRAINT ENGINEER_PK PRIMARY KEY (engID),
CONSTRAINT ENGINEER_FK FOREIGN KEY (spec) REFERENCES PRODUCT(part_no));
```

Column Name	Data Type	Nullable	Default	Primary Key
ENGID	NUMBER(10,0)	No	-	1
FNAME	NVARCHAR2(30)	Yes	-	-
LNAME	NVARCHAR2(30)	Yes	-	-
TEL_NO	VARCHAR2(15)	Yes	-	-
EMAILID	NVARCHAR2(30)	Yes	-	-
ADDRESS	NVARCHAR2(50)	Yes	-	-
SPEC	NUMBER(10,0)	Yes	-	-

1 - 7

Data insertion

EDIT	ENGID	FNAME	LNAME	TEL_NO	EMAILID	ADDRESS	SPEC
	300	Martin	Specs	0882341279	martin.specs@mycomp.ie	31 Baggot Street	295
	301	Freddie	Oswald	0832341123	freddie.oswald@mycomp.ie	25 James Street	101
	302	James	McArthur	0834356901	james.mcarthur@mycomp.ie	15 King George st	125
	303	John	Bain	0813245890	john.bain@mycomp.ie	87 Turvey Avenue	175
	304	Michael	Oshea	0876543290	michael.oshea@mycomp.ie	15 Temple Bar	188
	305	James	Franco	0832456782	james.franco@mycomp.ie	18 Limited Road	190
	306	Matthew	Elmer	0878945439	matt.elmer@mycomp.ie	91 Emmet Road	320
	307	Frank	Oneill	0875643210	frank.oneill@mycomp.ie	21 Circular Road	-
	308	Jim	Sullivan	088234190	jim.sullivan@mycomp.ie	85 Leicester Street	-
	309	Robert	Duff	0823217890	rodert.duff@mycomp.ie	19 Suffolk Road	-
	320	Mark	Brothon	0843657841	mark.brothon@mycomp.ie	13 Fitzwilliam Road	-
	321	Fergus	Nicol	0842563748	fergus.nicol@mycomp.ie	10 Turvey Avenue	-

row(s) 1 - 12 of 12

```
-- ONSITE_SUPPORT table creation
CREATE TABLE ONSITE_SUPPORT(
supportID NUMBER(10) NOT NULL,
customerID NUMBER(15) NOT NULL,
engID NUMBER(10) NOT NULL,
time TIMESTAMP,
duration NUMBER(2),
tcost NUMBER(10, 2),
CONSTRAINT ONSITE_SUPPORT_PK PRIMARY KEY (supportID),
CONSTRAINT ONSITE_SUPPORT_FK FOREIGN KEY (customerID) REFERENCES
CUSTOMER(customerID),
CONSTRAINT ONSITE_SUPPORT_FK2 FOREIGN KEY (engID) REFERENCES ENGINEER(engID));
```

Column Name	Data Type	Nullable	Default	Primary Key
SUPPORTID	NUMBER(10,0)	No	-	1
CUSTOMERID	NUMBER(15,0)	No	-	-
ENGID	NUMBER(10,0)	No	-	-
TIME	TIMESTAMP(6)	Yes	-	-
DURATION	NUMBER(2,0)	Yes	-	-
TCOST	NUMBER(10,2)	Yes	-	-

1 - 6

Data insertion

EDIT	SUPPORTID	CUSTOMERID	ENGID	TIME	DURATION	TCOST
	10000	517	302	14-FEB-14 09.30.00.000000 AM	2	400
	10001	506	309	09-MAR-14 11.00.00.000000 AM	3	650
	10002	520	300	10-MAR-14 08.30.00.000000 AM	6	1450
	10003	517	308	14-MAR-14 12.30.00.000000 PM	2	450
	10004	509	301	14-MAR-14 10.00.00.000000 AM	6	1500
	10005	510	303	19-MAR-14 09.00.00.000000 AM	4	950
	10006	517	305	21-MAR-14 11.30.00.000000 AM	1	250
	10007	507	304	23-MAR-14 10.30.00.000000 AM	3	650
	10008	511	306	27-MAR-14 02.30.00.000000 PM	1	250
	10009	508	307	29-MAR-14 09.00.00.000000 AM	7	1800
	10010	515	308	04-APR-14 10.00.00.000000 AM	2	450
	10011	513	301	04-APR-14 08.30.00.000000 AM	5	1200
	10012	515	300	07-APR-14 10.30.00.000000 AM	5	1350
	10013	507	309	09-APR-14 12.30.00.000000 PM	4	950
	10014	509	304	11-APR-14 09.00.00.000000 AM	3	700

row(s) 1 - 15 of 17

```
-- ORDER_CALL table creation
CREATE TABLE ORDER_CALL(
orderID NUMBER(10) NOT NULL,
order_date DATE NOT NULL,
type NUMBER(1),
start_date DATE,
end_date DATE,
discount NUMBER(3),
contactID NUMBER(15) NOT NULL,
CONSTRAINT ORDER_CALL_PK PRIMARY KEY (orderID),
CONSTRAINT ORDER_CALL_FK FOREIGN KEY (contactID) REFERENCES
CONTACT_PERSON(contactID));
```

Column Name	Data Type	Nullable	Default	Primary Key
ORDERID	NUMBER(10,0)	No	-	1
ORDER_DATE	DATE	No	-	-
TYPE	NUMBER(1,0)	Yes	-	-
START_DATE	DATE	Yes	-	-
END_DATE	DATE	Yes	-	-
DISCOUNT	NUMBER(3,0)	Yes	-	-
CONTACTID	NUMBER(15,0)	No	-	-

1 - 7

Data insertion

EDIT	ORDERID	ORDER_DATE	TYPE	START_DATE	END_DATE	DISCOUNT	CONTACTID
	100040	04/03/2012	3	06/01/2012	06/01/2014	3	1502
	100041	04/03/2013	3	05/01/2013	05/01/2014	0	1514
	100042	04/21/2013	3	05/01/2013	05/31/2014	0	1513
	100043	04/03/2014	3	05/01/2014	04/30/2018	4	1502
	100044	04/03/2014	1	06/01/2014	06/01/2016	3	1500
	100045	04/04/2014	1	05/01/2014	05/01/2015	0	1514
	100046	04/07/2014	2	-	-	-	1511
	100047	04/08/2014	1	09/01/2014	09/01/2018	5	1500
	100048	04/09/2014	3	06/01/2014	06/01/2015	0	1503
	100049	04/09/2014	3	07/01/2014	09/30/2014	0	1508
	100050	04/10/2014	2	-	-	-	1505
	100051	04/11/2014	1	05/01/2014	05/01/2020	7	1507

row(s) 1 - 12 of 12

```
-- ORDER_LINE table creation
CREATE TABLE ORDER_LINE(
orderID NUMBER(10) NOT NULL,
part_no NUMBER(10) NOT NULL,
qty NUMBER(5) NOT NULL,
item_price NUMBER(10, 2) NOT NULL,
CONSTRAINT ORDER_LINE_PK PRIMARY KEY (orderID, part_no),
CONSTRAINT ORDER_LINE_FK FOREIGN KEY (orderID) REFERENCES ORDER_CALL(orderID),
CONSTRAINT ORDER_LINE_FK2 FOREIGN KEY (part_no) REFERENCES PRODUCT(part_no));
```

Column Name	Data Type	Nullable	Default	Primary Key
ORDERID	NUMBER(10,0)	No	-	1
PART_NO	NUMBER(10,0)	No	-	2
QTY	NUMBER(5,0)	No	-	-
ITEM_PRICE	NUMBER(10,2)	No	-	-

1 - 4

Data insertion

EDIT	ORDERID	PART_NO	QTY	ITEM_PRICE
	100049	295	5	75
	100046	295	5	1800
	100040	101	3	50.5
	100040	175	2	15.85
	100041	860	4	85.99
	100042	190	5	65
	100042	101	1	50.5
	100043	520	5	125
	100044	521	2	45.95
	100045	175	5	15.85
	100045	320	5	80
	100047	188	2	45
	100048	101	4	50.5
	100050	295	3	1800
	100050	320	3	1250

```
-- Order TYPE table creation - this is only a lookup table (Look up table)
CREATE TABLE TYPE(
typeID NUMBER(1),
name NVARCHAR2(15),
CONSTRAINT TYPE_PK PRIMARY KEY (typeID));
```

Column Name	Data Type	Nullable	Default	Primary Key
TYPEID	NUMBER(1,0)	No	-	1
NAME	NVARCHAR2(15)	Yes	-	-
				1 - 2

Data insertion

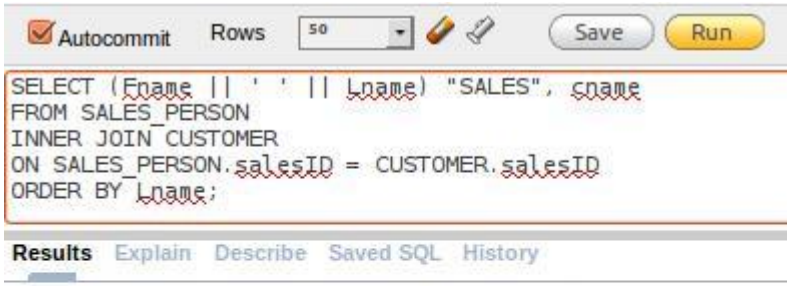
EDIT	TYPEID	NAME
	1	Quote
	2	Order
	3	Support
		row(s) 1 - 3 of 3

```
-- Junction table creation – it has not been used or actually created in the database
CREATE TABLE JUNCTION(
part_no NUMBER(15),
engID NUMBER(15),
CONSTRAINT JUNCTION_PK PRIMARY KEY (part_no,engID),
CONSTRAINT JUNCTION_FK FOREIGN KEY (part_no) REFERENCES PRODUCT(part_no),
CONSTRAINT JUNCTION_FK2 FOREIGN KEY (engID) REFERENCES ENGINEER(engID));
```

2. 4 INNER JOIN queries with descriptions

Q2.1 The sales manager wants to see which sales employee is assigned to which customer

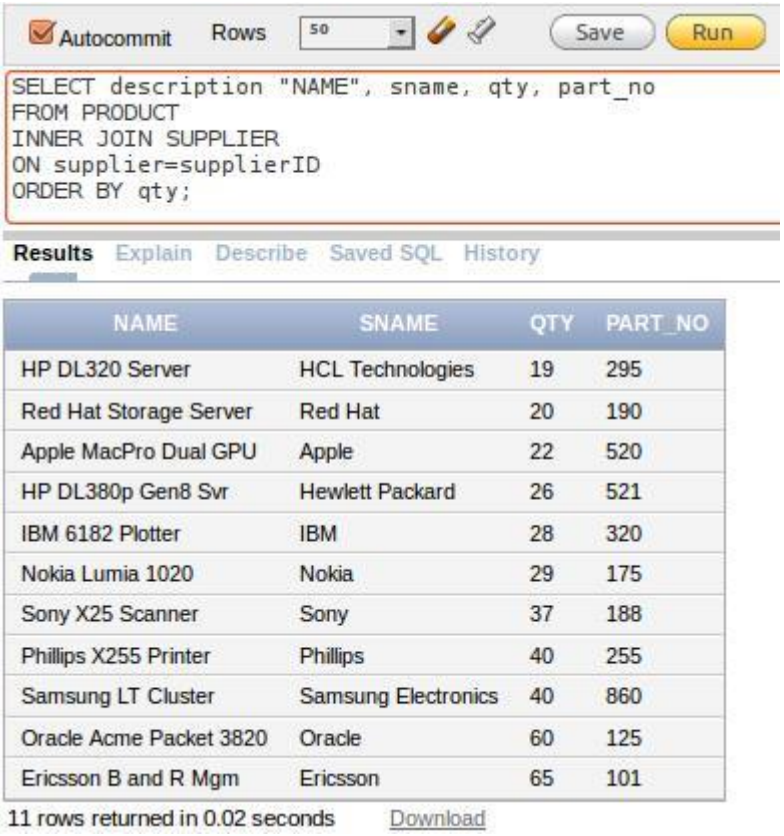
```
SELECT (Fname || ' ' || Lname) "SALES", cname
FROM SALES_PERSON
INNER JOIN CUSTOMER
ON SALES_PERSON.salesID = CUSTOMER.salesID
ORDER BY Lname;
```



SALES	CNAME
Henry Cavil	IPA
Henry Cavil	Irish Pride
Henry Cavil	Oxfam
Henry Cavil	IRFU
Henry Cavil	Layca
Henry Cavil	AOL
Henry Cavil	Technologies
Henry Cavil	Travel Europe
Jon Gil	UCD
Jon Gil	Stores
Jon Gil	Snickers
Jon Gil	Cinemax
Jon Gil	Hewlett Packard
Jon Gil	Fiverr
Jill Hart	France Tel
Jill Hart	Masters
Jill Hart	Film4
Jill Hart	Form4
Jill Hart	Fasttrack

-- Q2.2 The Logistic department wants to see what is the current product quantity for each product and its supplier

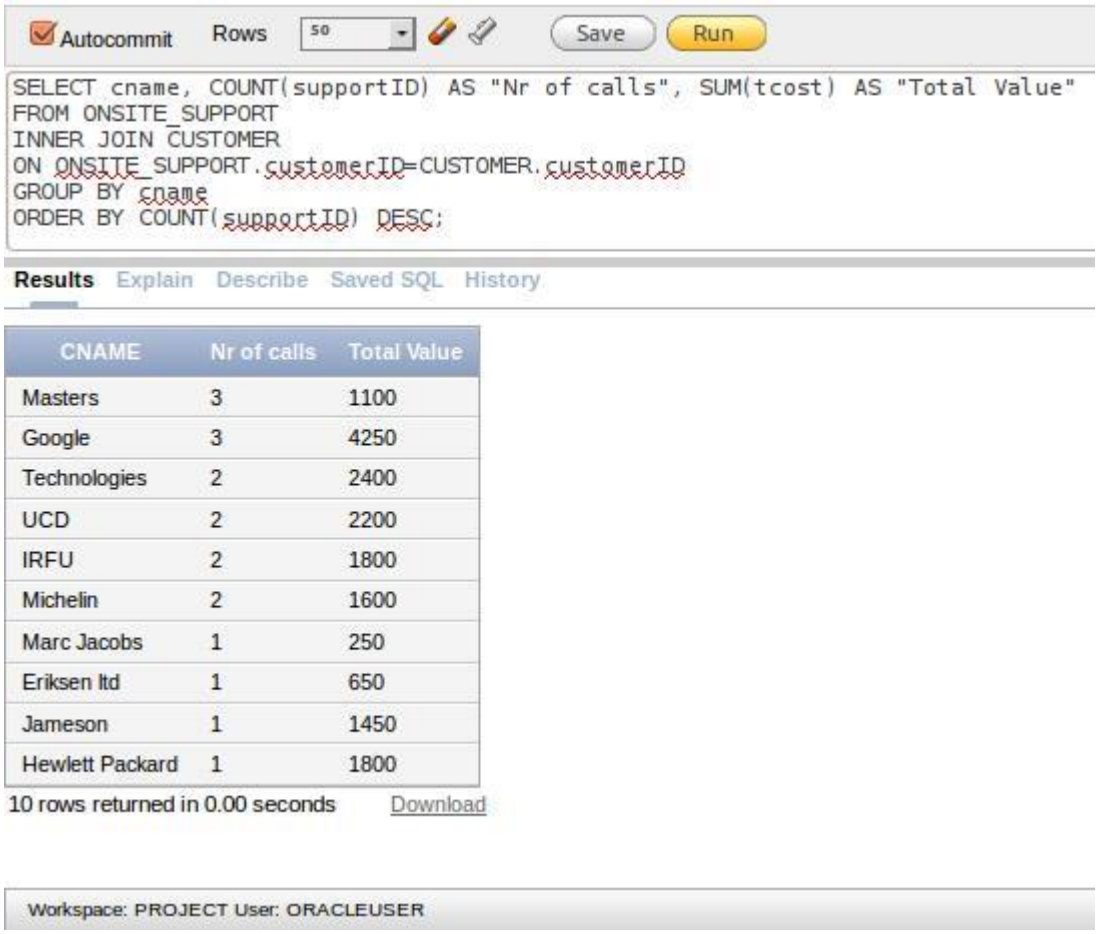
```
SELECT description "NAME", sname, qty, part_no
FROM PRODUCT
INNER JOIN SUPPLIER
ON supplier=supplierID
ORDER BY qty;
```



NAME	SNAME	QTY	PART_NO
HP DL320 Server	HCL Technologies	19	295
Red Hat Storage Server	Red Hat	20	190
Apple MacPro Dual GPU	Apple	22	520
HP DL380p Gen8 Svr	Hewlett Packard	26	521
IBM 6182 Plotter	IBM	28	320
Nokia Lumia 1020	Nokia	29	175
Sony X25 Scanner	Sony	37	188
Phillips X255 Printer	Phillips	40	255
Samsung LT Cluster	Samsung Electronics	40	860
Oracle Acme Packet 3820	Oracle	60	125
Ericsson B and R Mgm	Ericsson	65	101

11 rows returned in 0.02 seconds Download

```
-- Q2.3 The sales manager wants to see which are the customers that have
required onsite support (which is different than the contract_support) so
that he can approach those customers and propose them support contracts as
a more convenient solution
SELECT cname, COUNT(supportID) AS "Nr of calls", SUM(tcost) AS "Total
Value"
FROM ONSITE_SUPPORT
INNER JOIN CUSTOMER
ON ONSITE_SUPPORT.customerID=CUSTOMER.customerID
GROUP BY cname
ORDER BY COUNT(supportID) DESC;
```



CNAME	Nr of calls	Total Value
Masters	3	1100
Google	3	4250
Technologies	2	2400
UCD	2	2200
IRFU	2	1800
Michelin	2	1600
Marc Jacobs	1	250
Eriksen ltd	1	650
Jameson	1	1450
Hewlett Packard	1	1800

10 rows returned in 0.00 seconds Download

```
SELECT orderID AS
   "Quote nr", CONTACT_PERSON.cname "Contact", CONTACT_PERSON.tel_no "Tel number",
CONTACT_PERSON.emailID "Email", CUSTOMER.cname "Customer", (SALES_PERSON.Fname || ' '||
SALES_PERSON.Lname) "Sales", SALES_PERSON.emailID "Email"
   FROM CONTACT_PERSON
   INNER JOIN ORDER_CALL
   ON ORDER_CALL.contactID=CONTACT_PERSON.contactID
   INNER JOIN CUSTOMER  -- needs to join the customer table to get the sales person assigned to it
   ON CUSTOMER.customerID=CONTACT_PERSON.customerID
   INNER JOIN SALES_PERSON     -- gets the sales id
   ON CUSTOMER.salesID=SALES_PERSON.salesID
   WHERE type=1;
```

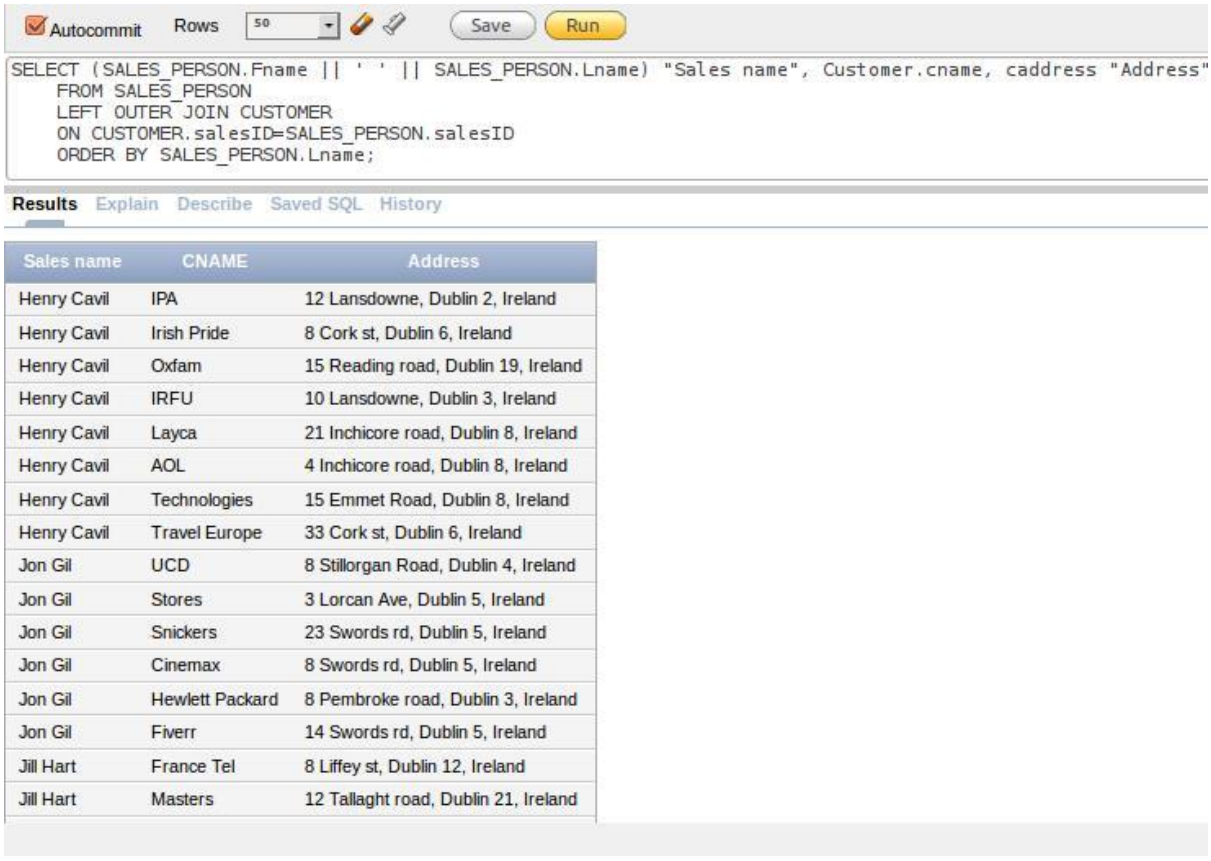
Autocommit Rows 50 Save Run

Results Explain Describe Saved SQL History

Quote nr	Contact	Tel number	Email	Customer	Sales	Email
100047	Alan Rogers	0981232149	alan_rogers@erikssen.ie	Eriksen ltd	Eoin McBeth	eoin_mcbeth@mycomp.com
100044	Alan Rogers	0981232149	alan_rogers@erikssen.ie	Eriksen ltd	Eoin McBeth	eoin_mcbeth@mycomp.com
100204	Marc Ohara	0865431789	marc_ohara@hp.com	Hewlett Packard	Jon Gil	jon_gil@mycomp.com
100183	Alan Pardew	0831256123	a_pardew@jacobs.ie	Marc Jacobs	Maria Heart	maria_heart@mycomp.com
100217	Mary Poppins	0831256123	m_poppins@tech.ie	Technologies	Henry Cavil	henry_cavil@mycomp.com
100144	Mary Poppins	0831256123	m_poppins@tech.ie	Technologies	Henry Cavil	henry_cavil@mycomp.com
100078	Mary Poppins	0831256123	m_poppins@tech.ie	Technologies	Henry Cavil	henry_cavil@mycomp.com
100051	Mary Poppins	0831256123	m_poppins@tech.ie	Technologies	Henry Cavil	henry_cavil@mycomp.com
100189	Grant Mar	0831123678	gmar@irfu.ie	IRFU	Henry Cavil	henry_cavil@mycomp.com
100173	Grant Mar	0831123678	gmar@irfu.ie	IRFU	Henry Cavil	henry_cavil@mycomp.com
100194	Niamh Brothar	0835654578	n.brothar@laser.ie	Laserprint	Frank Ohara	frank_ohara@mycomp.com
100209	James Joyce	0831123888	jjoyce@masters.ie	Masters	Jill Hart	jill_hart@mycomp.com

3. 6 OUTER JOIN (2 x left, 2 x full, 2 x right) queries with descriptions

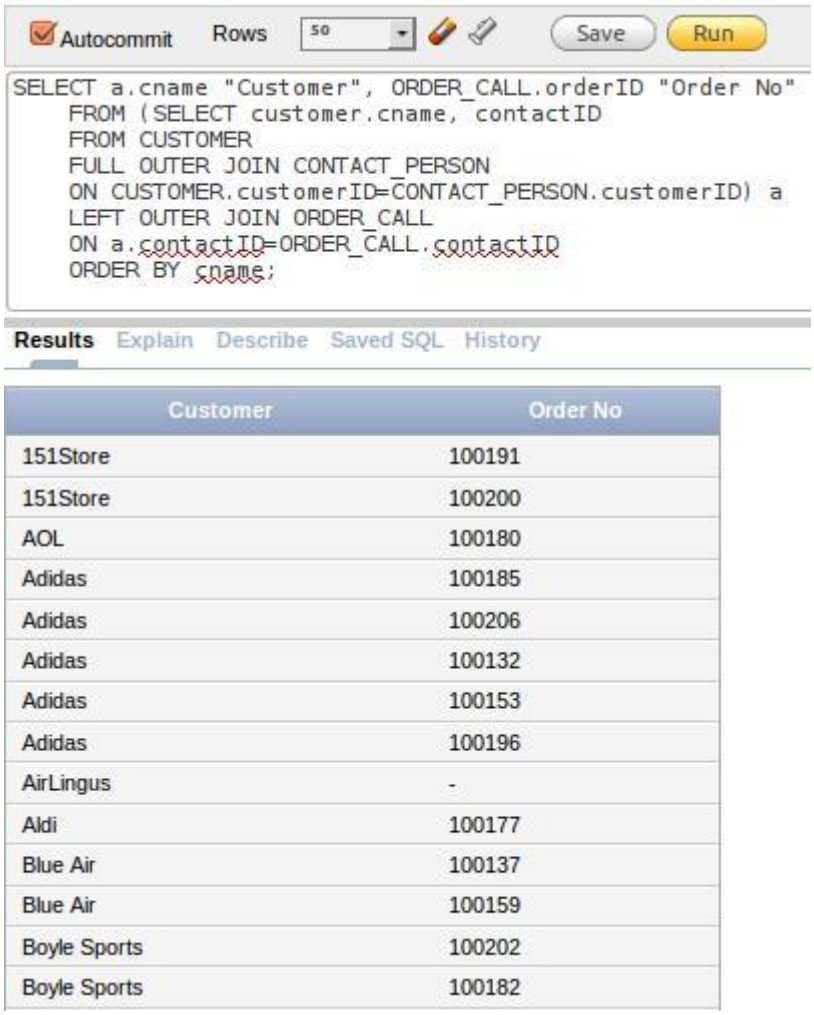
```
-- LEFT OUTER JOIN
-- Q3.1 The sales manager wants to know the customers allocated to each
sales person with their address so the sales can pay them a visit for a
meeting
SELECT (SALES_PERSON.Fname || ' ' || SALES_PERSON.Lname) "Sales name", Customer.cname,
caddress "Address"
   FROM SALES_PERSON
   LEFT OUTER JOIN CUSTOMER
   ON CUSTOMER.salesID=SALES_PERSON.salesID
   ORDER BY SALES_PERSON.Lname;
```



Sales name	CNAME	Address
Henry Cavil	IPA	12 Lansdowne, Dublin 2, Ireland
Henry Cavil	Irish Pride	8 Cork st, Dublin 6, Ireland
Henry Cavil	Oxfam	15 Reading road, Dublin 19, Ireland
Henry Cavil	IRFU	10 Lansdowne, Dublin 3, Ireland
Henry Cavil	Layca	21 Inchicore road, Dublin 8, Ireland
Henry Cavil	AOL	4 Inchicore road, Dublin 8, Ireland
Henry Cavil	Technologies	15 Emmet Road, Dublin 8, Ireland
Henry Cavil	Travel Europe	33 Cork st, Dublin 6, Ireland
Jon Gil	UCD	8 Stillorgan Road, Dublin 4, Ireland
Jon Gil	Stores	3 Lorcan Ave, Dublin 5, Ireland
Jon Gil	Snickers	23 Swords rd, Dublin 5, Ireland
Jon Gil	Cinemax	8 Swords rd, Dublin 5, Ireland
Jon Gil	Hewlett Packard	8 Pembroke road, Dublin 3, Ireland
Jon Gil	Fiverr	14 Swords rd, Dublin 5, Ireland
Jill Hart	France Tel	8 Liffey st, Dublin 12, Ireland
Jill Hart	Masters	12 Tallaght road, Dublin 21, Ireland

-- Q3.2 A manager wants to know which are the customers that have made at least one order call (have ordered a quote, a support contract, or have made an order) and the customers that have not made an order call

```
SELECT a.cname "Customer", ORDER_CALL.orderID "Order No"
  FROM (SELECT customer.cname, contactID
  FROM CUSTOMER
  FULL OUTER JOIN CONTACT_PERSON
  ON CUSTOMER.customerID=CONTACT_PERSON.customerID) a
  LEFT OUTER JOIN ORDER_CALL
  ON a.contactID=ORDER_CALL.contactID
  ORDER BY cname;
```

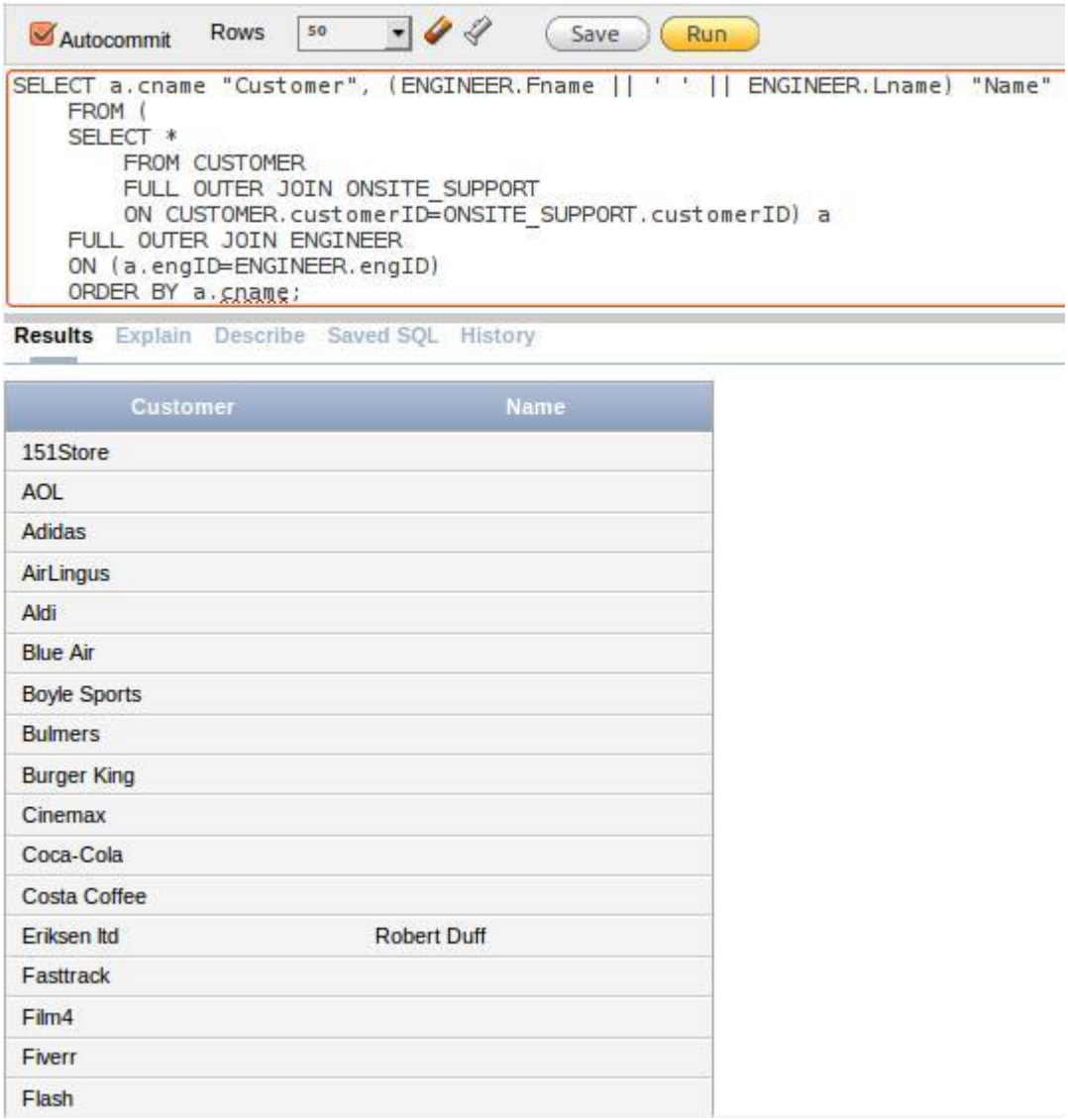


Customer	Order No
151Store	100191
151Store	100200
AOL	100180
Adidas	100185
Adidas	100206
Adidas	100132
Adidas	100153
Adidas	100196
AirLingus	-
Aldi	100177
Blue Air	100137
Blue Air	100159
Boyle Sports	100202
Boyle Sports	100182

Flash	100212
Form4	-
Foster	100143
Four Star	-
FoxPro	-
France Tel	-
Google	100213
Heineken	100082
Heineken	100049
Heineken	100184
Heirtz	100130
Heirtz	100126
Heirtz	100129
Heirtz	100219
Heirtz	100199
Hewlett Packard	100040
Hewlett Packard	100043
More than 50 rows available. Increase rows selector to view more rows.	

50 rows returned in 0.01 seconds Download

```
-- FULL OUTER JOIN
-- Q3.3 A query is needed to see which engineers have visited different sales for onsite support services, and which are the customers that have not requested onsite support services and the engineers that have not visited customers for onsite support calls
SELECT a.cname "Customer", (ENGINEER.Fname || ' ' || ENGINEER.Lname) "Name"
  FROM (
  SELECT *
    FROM CUSTOMER
    FULL OUTER JOIN ONSITE_SUPPORT
    ON CUSTOMER.customerID=ONSITE_SUPPORT.customerID) a
  FULL OUTER JOIN ENGINEER
  ON (a.engID=ENGINEER.engID)
  ORDER BY a.cname;
```

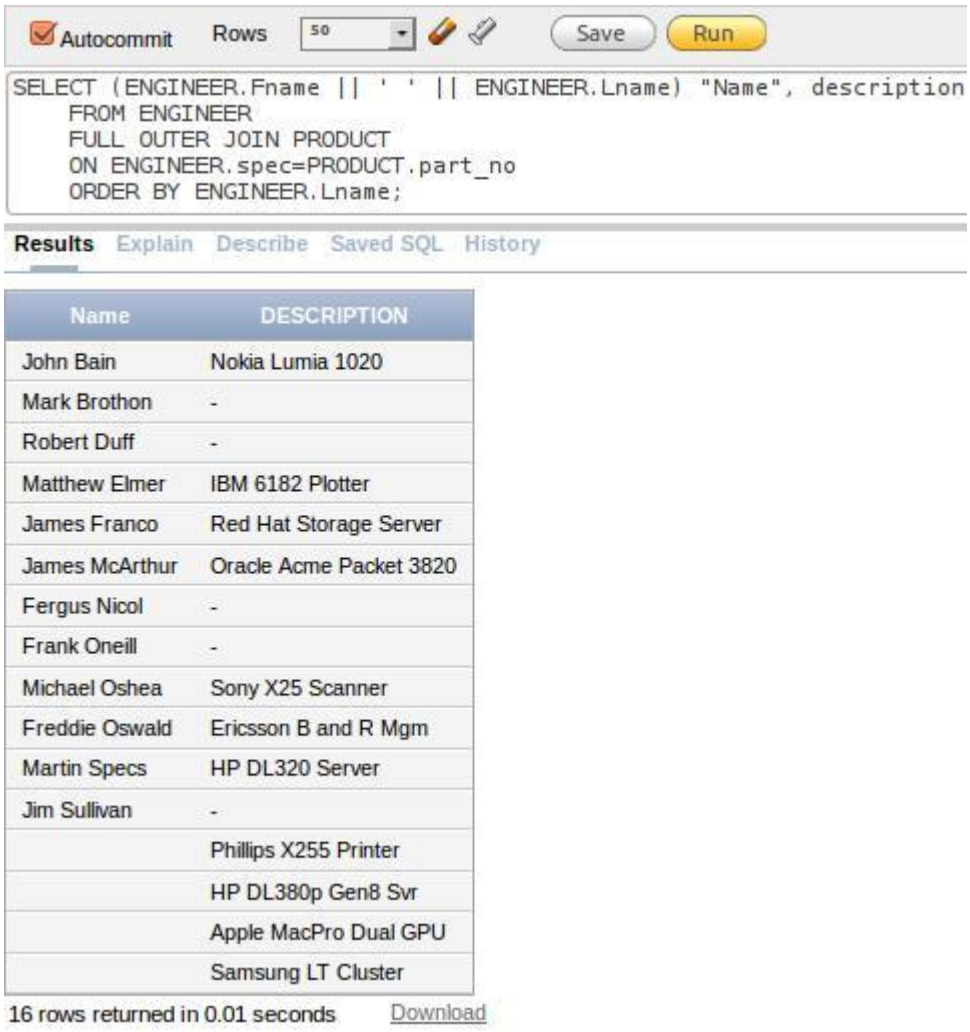


Customer	Name
151Store	
AOL	
Adidas	
AirLingus	
Aldi	
Blue Air	
Boyle Sports	
Bulmers	
Burger King	
Cinemax	
Coca-Cola	
Costa Coffee	
Eriksen ltd	Robert Duff
Fasttrack	
Film4	
Fiverr	
Flash	

Foster	
Four Star	
FoxPro	
France Tel	
Google	Matthew Elmer
Google	Matthew Elmer
Google	John Bain
Heineken	
Heirtz	
Hewlett Packard	Frank Oneill
IPA	
IRFU	Martin Specs
IRFU	Jim Sullivan
Irish Pride	
JD Sports	
James.ie	
Jameson	Martin Specs
John West	
KFC	
Laserprint	
Layca	
Lidl	
Lifestyle	
Marc Jacobs	Matthew Elmer
Masters	James Franco
Masters	James McArthur
Masters	Jim Sullivan
May West	

-- Q3.4 A manager wants to see the engineers specialized in a type of product and what are the type of products that don't have an engineer specialized so he can send the rest to trainings for the rest of the type of products

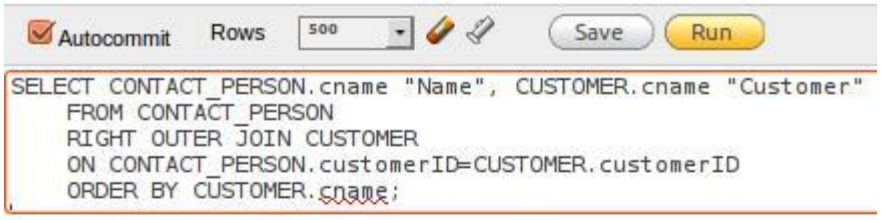
```
SELECT (ENGINEER.Fname || ' ' || ENGINEER.Lname) "Name", description
   FROM ENGINEER
   FULL OUTER JOIN PRODUCT
   ON ENGINEER.spec=PRODUCT.part_no
   ORDER BY ENGINEER.Lname;
```



Name	DESCRIPTION
John Bain	Nokia Lumia 1020
Mark Brothon	-
Robert Duff	-
Matthew Elmer	IBM 6182 Plotter
James Franco	Red Hat Storage Server
James McArthur	Oracle Acme Packet 3820
Fergus Nicol	-
Frank Oneill	-
Michael Oshea	Sony X25 Scanner
Freddie Oswald	Ericsson B and R Mgm
Martin Specs	HP DL320 Server
Jim Sullivan	-
	Phillips X255 Printer
	HP DL380p Gen8 Svr
	Apple MacPro Dual GPU
	Samsung LT Cluster

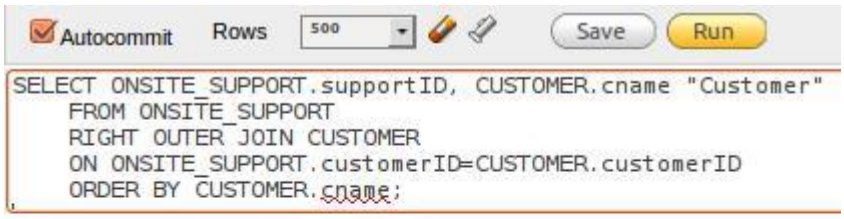
16 rows returned in 0.01 seconds Download

```
-- RIGHT OUTER JOIN
-- Q3.5 The sales manager wants to run a query to see which customer is a contact person assigned to and which are the customers that don't have a contact person in the database
SELECT CONTACT_PERSON.cname "Name", CUSTOMER.cname "Customer"
   FROM CONTACT_PERSON
   RIGHT OUTER JOIN CUSTOMER
   ON CONTACT_PERSON.customerID=CUSTOMER.customerID
   ORDER BY CUSTOMER.cname;
```



Name	Customer
Otto Bradley	151Store
Luke York	AOL
Neil Borg	Adidas
Cody Matthews	Adidas
Forrest Shelton	AirLingus
Ishmael Booker	Aldi
Cedric Watkins	Blue Air
Castor Mccoy	Boyle Sports
Ignatius Wright	Bulmers
Yoshio Hahn	Burger King
George Rodgers	Cinemax
Galvin Gates	Coca-Cola
Avram Lane	Costa Coffee
Alan Rogers	Eriksen ltd
Declan Lynn	Fasttrack

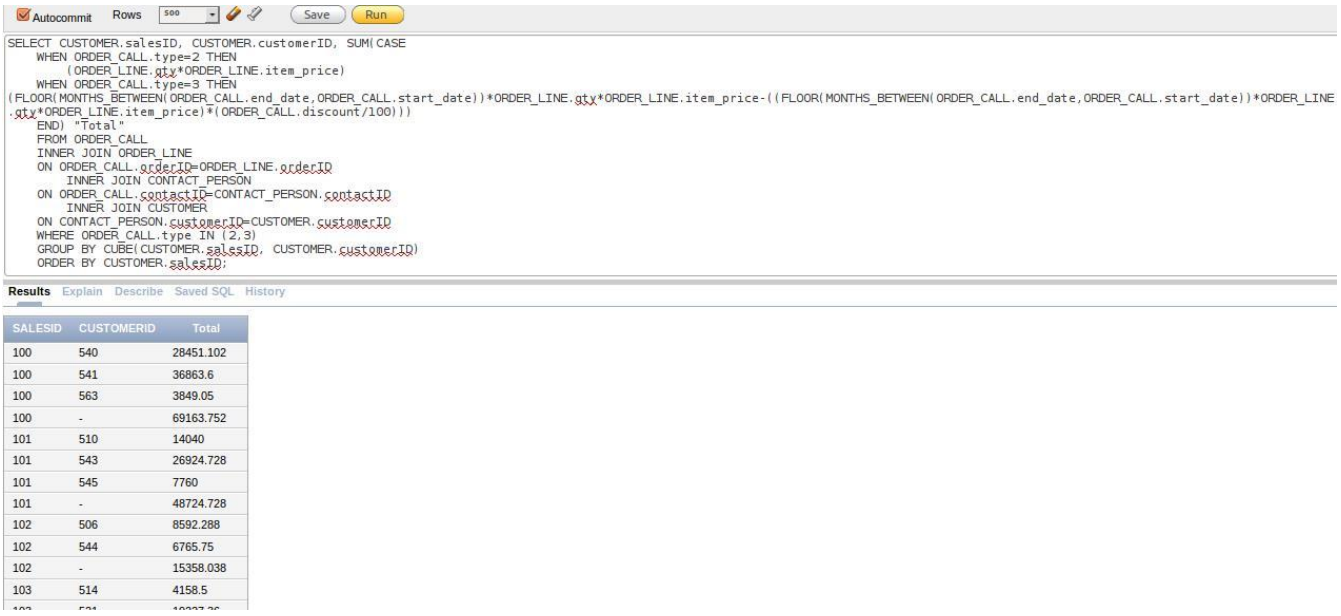
```
-- Q3.6 A manager needs a query to retrieve the customers that have requested onsite support services and which are those
SELECT ONSITE_SUPPORT.supportID, CUSTOMER.cname "Customer"
   FROM ONSITE_SUPPORT
   RIGHT OUTER JOIN CUSTOMER
   ON ONSITE_SUPPORT.customerID=CUSTOMER.customerID
   ORDER BY CUSTOMER.cname;
```



SUPPORTID	Customer
-	151Store
-	AOL
-	Adidas
-	AirLingus
-	Aldi
-	Blue Air
-	Boyle Sports
-	Bulmers
-	Burger King
-	Cinemax
-	Coca-Cola
-	Costa Coffee
10001	Eriksen ltd
-	Fasttrack
-	Film4

4. 1 CUBE query (with at least 2 columns)

```
-- Q4 CUBE The sales manager wants to see a report for each sales that has been involved in a sale or order of a support contract with with the value of it and total values per customers
SELECT CUSTOMER.salesID, CUSTOMER.customerID, SUM(CASE
   WHEN ORDER_CALL.type=2 THEN  -- if it is only a sales order, the price is calculated based on item price multiplied by qty
      (ORDER_LINE.qty*ORDER_LINE.item_price)
   WHEN ORDER_CALL.type=3 THEN -- if it is a contract support it needs to check if the discount is needed
(FLOOR(MONTHS_BETWEEN(ORDER_CALL.end_date,ORDER_CALL.start_date))*ORDER_LINE.qty*ORDER_LINE.item_price-
((FLOOR(MONTHS_BETWEEN(ORDER_CALL.end_date,ORDER_CALL.start_date))*ORDER_LINE.qty*ORDER_LINE.item_price)*(ORDER_CALL.discount/100)))
-- floor rounds to check the number of months and discount; the formula calculates the total price
   END) "Total"
   FROM ORDER_CALL
   INNER JOIN ORDER_LINE
   ON ORDER_CALL.orderID=ORDER_LINE.orderID
      INNER JOIN CONTACT_PERSON
   ON ORDER_CALL.contactID=CONTACT_PERSON.contactID
      INNER JOIN CUSTOMER
   ON CONTACT_PERSON.customerID=CUSTOMER.customerID
   WHERE ORDER_CALL.type IN (2,3)
   GROUP BY CUBE(CUSTOMER.salesID, CUSTOMER.customerID)
   ORDER BY CUSTOMER.salesID;
```

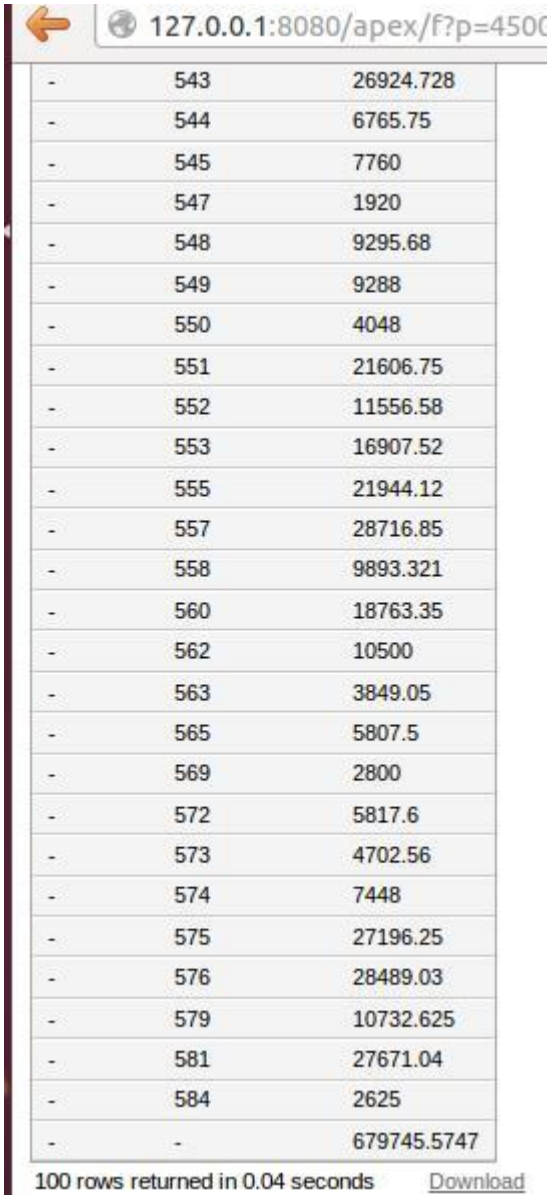


Autocommit Rows 500 Save Run

```
SELECT CUSTOMER.salesID, CUSTOMER.customerID, SUM(CASE
    WHEN ORDER_CALL.type=2 THEN
        (ORDER_LINE.qty*ORDER_LINE.item_price)
    WHEN ORDER_CALL.type=3 THEN
(FLOOR(MONTHS_BETWEEN(ORDER_CALL.end_date,ORDER_CALL.start_date))*ORDER_LINE.qty*ORDER_LINE.item_price-((FLOOR(MONTHS_BETWEEN(ORDER_CALL.end_date,ORDER_CALL.start_date))*ORDER_LINE
.qty*ORDER_LINE.item_price)*(ORDER_CALL.discount/100)))
    END) "Total"
    FROM ORDER_CALL
    INNER JOIN ORDER_LINE
    ON ORDER_CALL.orderID=ORDER_LINE.orderID
        INNER JOIN CONTACT_PERSON
    ON ORDER_CALL.contactID=CONTACT_PERSON.contactID
        INNER JOIN CUSTOMER
    ON CONTACT_PERSON.customerID=CUSTOMER.customerID
    WHERE ORDER_CALL.type IN (2,3)
    GROUP BY CUBE(CUSTOMER.salesID, CUSTOMER.customerID)
    ORDER BY CUSTOMER.salesID;
```

Results Explain Describe Saved SQL History

SALESID	CUSTOMERID	Total
100	540	28451.102
100	541	36863.6
100	563	3849.05
100	-	69163.752
101	510	14040
101	543	26924.728
101	545	7760
101	-	48724.728
102	506	8592.288
102	544	6765.75
102	-	15358.038
103	514	4158.5



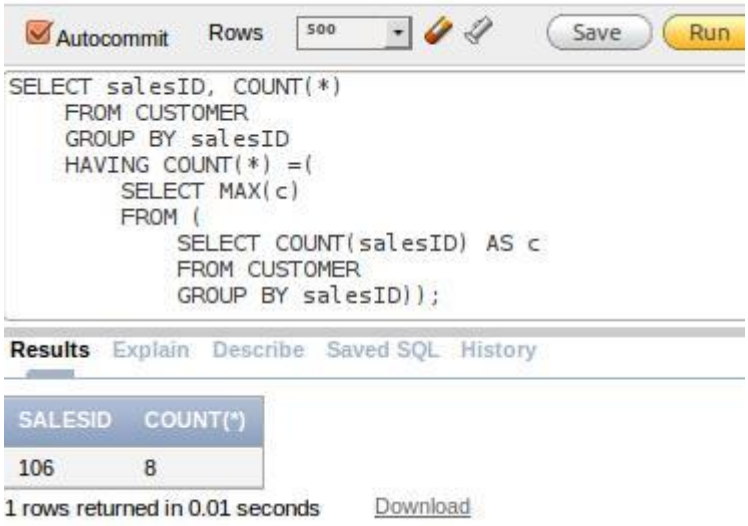
-	543	26924.728
-	544	6765.75
-	545	7760
-	547	1920
-	548	9295.68
-	549	9288
-	550	4048
-	551	21606.75
-	552	11556.58
-	553	16907.52
-	555	21944.12
-	557	28716.85
-	558	9893.321
-	560	18763.35
-	562	10500
-	563	3849.05
-	565	5807.5
-	569	2800
-	572	5817.6
-	573	4702.56
-	574	7448
-	575	27196.25
-	576	28489.03
-	579	10732.625
-	581	27671.04
-	584	2625
-	-	679745.5747

100 rows returned in 0.04 seconds Download

5. 5 examples of sub-queries

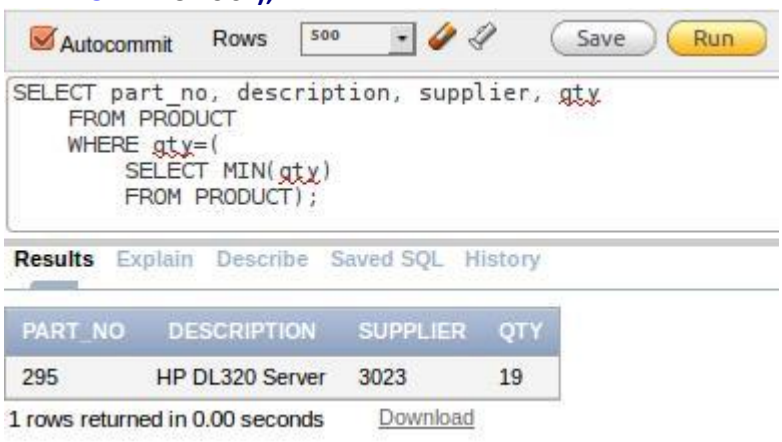
-- 1 A manager wants to see who is the sales that has the highest number of customers assigned so it won't assign new customers to the respective sales person until all the sales persons are looking after the same number of customers

```
SELECT salesID, COUNT(*)
   FROM CUSTOMER
   GROUP BY salesID
   HAVING COUNT(*) =(
      SELECT MAX(c)
      FROM (
         SELECT COUNT(salesID) AS c
         FROM CUSTOMER
         GROUP BY salesID));
```



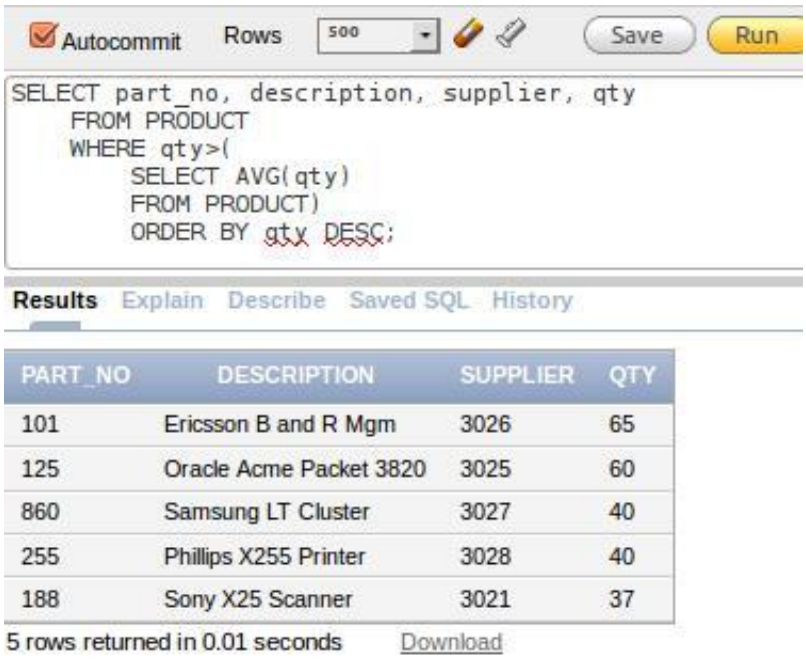
-- 2 There is a need to check which product has the minimum quantity to contact the supplier for a new purchase

```
SELECT part_no, description, supplier, qty
   FROM PRODUCT
   WHERE qty=(
      SELECT MIN(qty)
      FROM PRODUCT);
```



-- 3 A sales manager wants to see which are the products with quantity above the average, to ask the sales person to focust on selling those products

```
SELECT part_no, description, supplier, qty
   FROM PRODUCT
   WHERE qty>(
      SELECT AVG(qty)
      FROM PRODUCT)
      ORDER BY qty DESC;
```

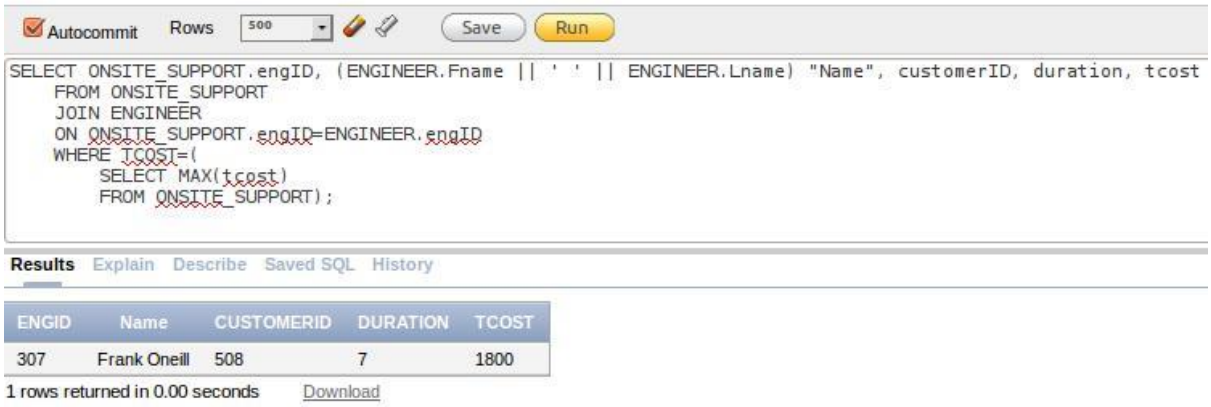


PART_NO	DESCRIPTION	SUPPLIER	QTY
101	Ericsson B and R Mgm	3026	65
125	Oracle Acme Packet 3820	3025	60
860	Samsung LT Cluster	3027	40
255	Phillips X255 Printer	3028	40
188	Sony X25 Scanner	3021	37

5 rows returned in 0.01 seconds Download

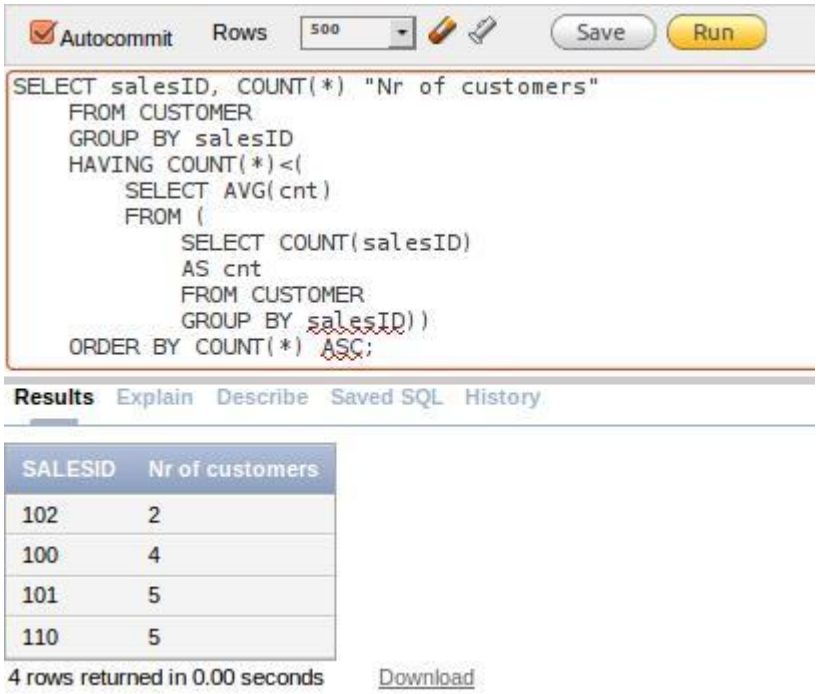
-- 4 A manager wants to see what engineer had generated the highest revenue on an onsite support visit, the value of it, the customer and the length

```
SELECT ONSITE_SUPPORT.engID, (ENGINEER.Fname || ' ' || ENGINEER.Lname) "Name", customerID, duration, tcost
   FROM ONSITE_SUPPORT
   JOIN ENGINEER
   ON ONSITE_SUPPORT.engID=ENGINEER.engID
   WHERE TCOST=(
      SELECT MAX(tcost)
      FROM ONSITE_SUPPORT);
```



-- 5 The sales manager wants to check which are the sales persons that have a number of customers allocated below the average number of customers allocated for each sales person so that he can assign new customers to those sales persons

```
SELECT salesID, COUNT(*) "Nr of customers"
   FROM CUSTOMER
   GROUP BY salesID
   HAVING COUNT(*)<(
      SELECT AVG(cnt)
      FROM (
         SELECT COUNT(salesID)
         AS cnt
         FROM CUSTOMER
         GROUP BY salesID))
   ORDER BY COUNT(*) ASC;
```

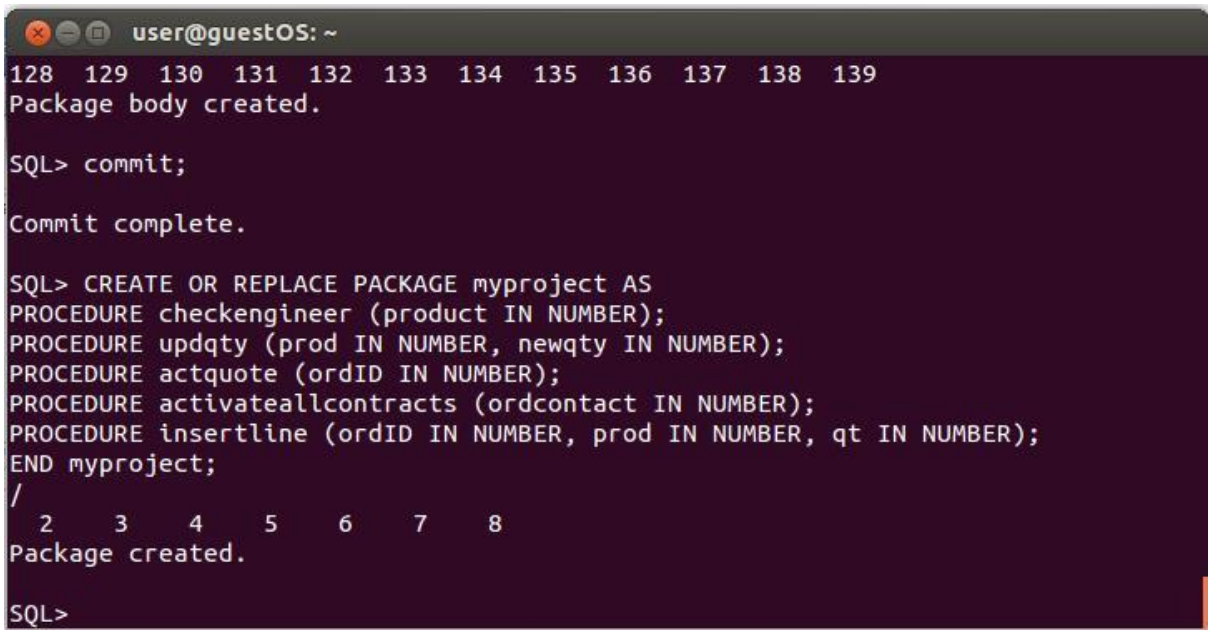


○

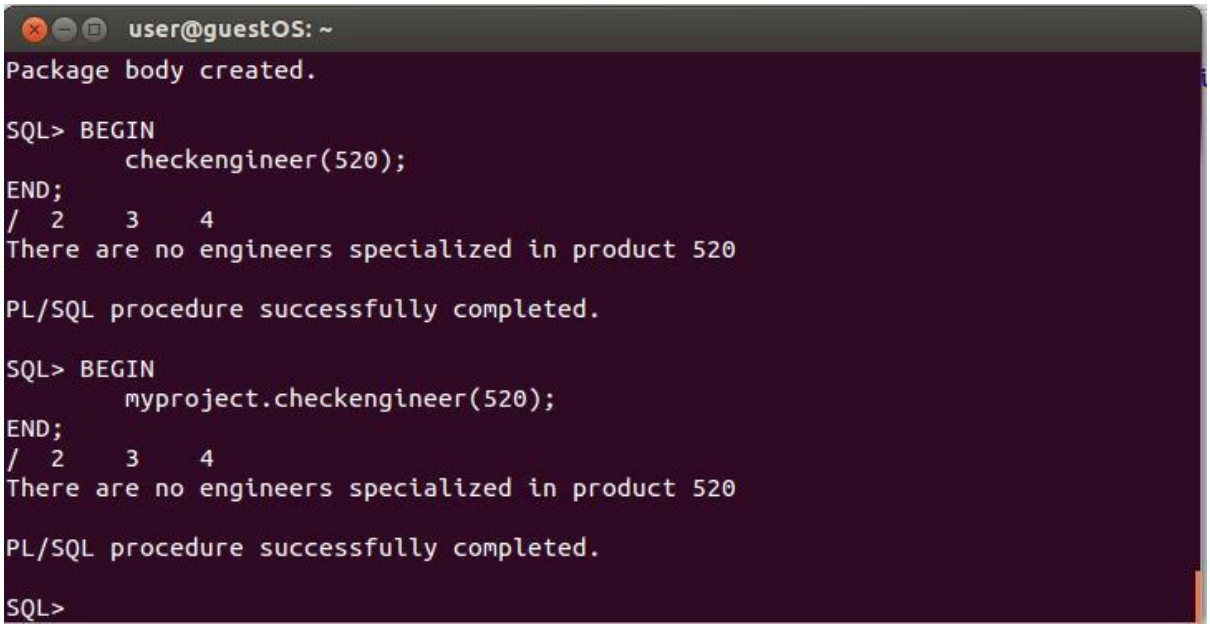
6. 5 PL/SQL procedures as part of one package. One procedure must demonstrate each of the following:

- **The use of a cursor**
- **The use of save points**
- **The use of a rollback**

```
CREATE OR REPLACE PACKAGE myproject AS
PROCEDURE checkengineer (product IN NUMBER);
PROCEDURE updqty (prod IN NUMBER, newqty IN NUMBER);
PROCEDURE actquote (ordID IN NUMBER);
PROCEDURE activateallcontracts (ordcontact IN NUMBER);
PROCEDURE insertline (ordID IN NUMBER, prod IN NUMBER, qt IN NUMBER);
END myproject;
/
```

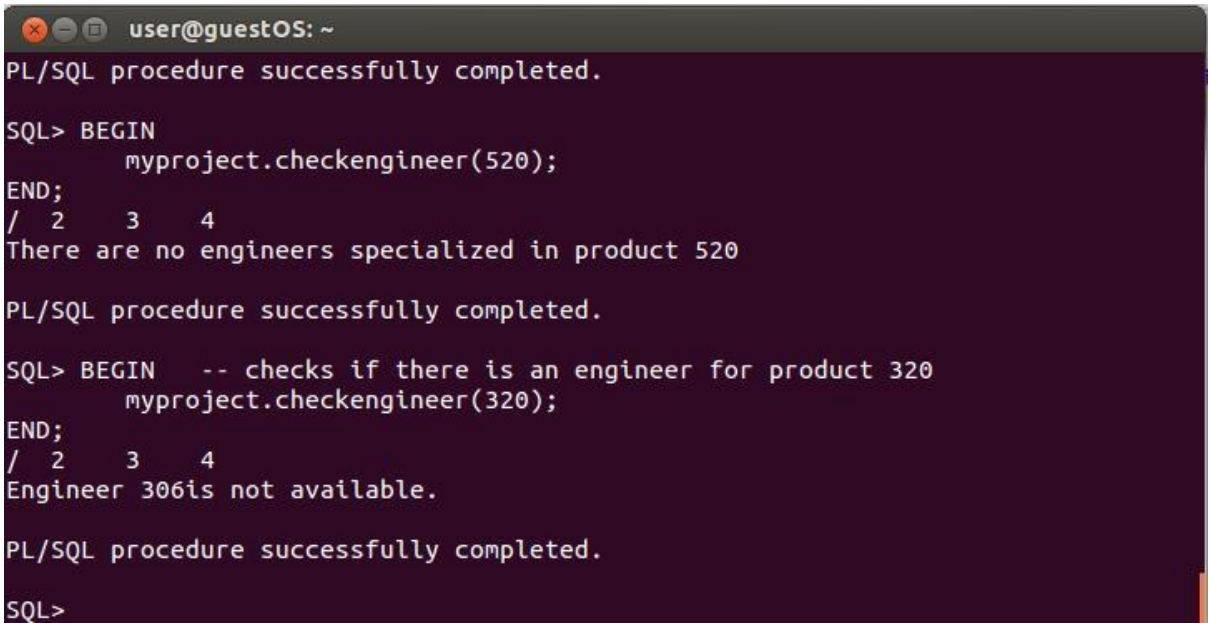


```
CREATE OR REPLACE PACKAGE BODY myproject
AS
-- Procedure that checks if there is an available engineer specialized in a specific product so that can might solve customer's problem quicker and it has experience. An engineer is not allowed to respond to more than one onsite support call during 7 consecituve days
PROCEDURE checkengineer (product IN NUMBER)
IS
engineer NUMBER(10);  -- stores the eng ID
available NUMBER(1);  -- checks if it has been on another site in the past 7 days
now DATE;  -- used for checking the nr of days between the current call and the last one for an engineer
CURSOR cur_check IS SELECT * FROM ONSITE_SUPPORT;  -- cursor that checks the onsite_support table
onsite_row cur_check%ROWTYPE;
BEGIN
available :=0;  -- 0 means an engineer is available
SELECT sysdate INTO now FROM DUAL;  -- assigns the current date to now
SELECT engID INTO ENGINEER FROM ENGINEER WHERE spec=product;  -- assigns the engID in the engineer ID
OPEN cur_check;
FETCH cur_check INTO onsite_row;  -- goes through each row
<<engineers>>
WHILE cur_check%FOUND LOOP   -- loop for going through each row
  IF onsite_row.engID=engineer AND (now-onsite_row.time)<7 THEN  -- checks if an engineer specialized in the given product exists and if it hasn't been on a site in the past 7 days
     available := available+1;
  END IF;
FETCH cur_check INTO onsite_row;  -- goes to the next row
END LOOP;
CLOSE cur_check;  -- closes the cursor; it has been through the whole table
IF available>0 THEN
  DBMS_OUTPUT.PUT_LINE('Engineer '|| engineer || 'is not available.');
ELSE
  DBMS_OUTPUT.PUT_LINE('Engineer '|| engineer || 'is available.');
END IF;
EXCEPTION  -- throughs this exception if there is no engID specialized in the given product
  WHEN NO_DATA_FOUND THEN
    DBMS_OUTPUT.PUT_LINE('There are no engineers specialized in product '|| product);
END checkengineer;
BEGIN  -- checks if there is an engineer for product 520 (no eng specialized in prod 520)
  myproject.checkengineer(520);
END;
/
```

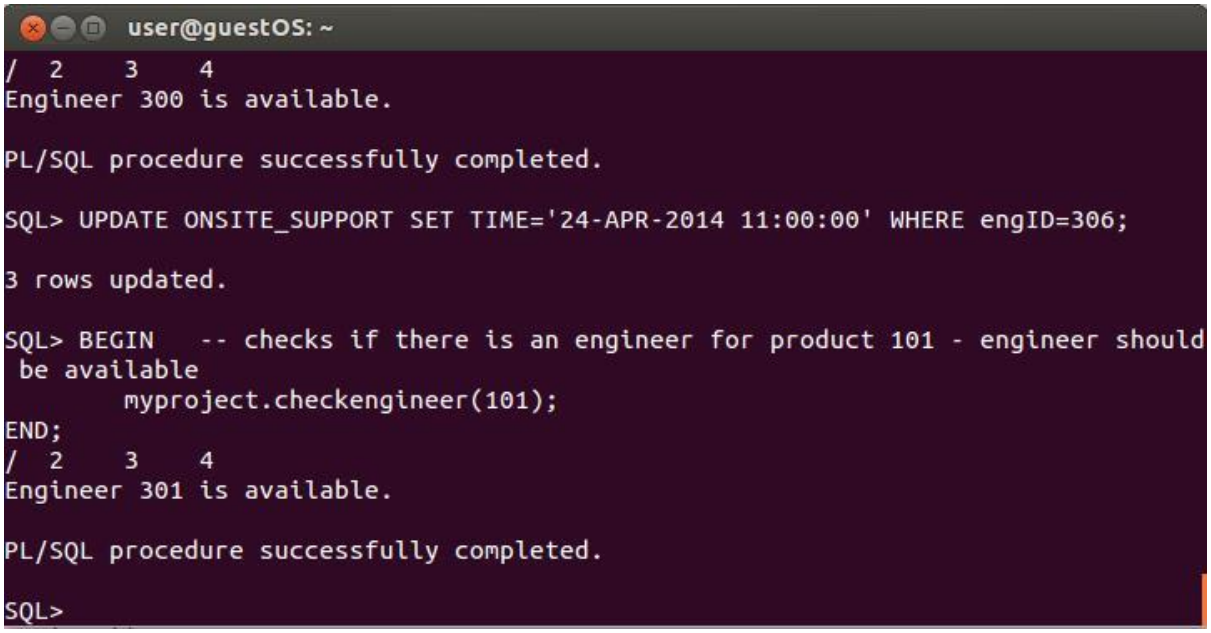


To check the procedure for an engineer that has already paid a visit to another customer in the past 7 days:

```
UPDATE ONSITE_SUPPORT SET TIME='24-APR-2014 11:00:00' WHERE engID=306;

BEGIN  -- checks if there is an engineer for product 320 - engineer should not be available because he has already been to an onsite call in the last 7 days
   myproject.checkengineer(320);
END;
/
```




```
BEGIN   -- checks if there is an engineer for product 101 - engineer should be available
    myproject.checkengineer(101);
END;
/
```




```
-- second procedure - updates the qty of a given product when the stock is being refilled
PROCEDURE updqty (prod IN NUMBER, newqty IN NUMBER)
IS
   qt NUMBER(4);
   BEGIN
   UPDATE PRODUCT SET    -- updates the qty with the old qty added to the new stock
   product.qty=product.qty+newqty
   WHERE part_no=prod;
   SELECT qty INTO qt FROM PRODUCT WHERE part_no=prod;
   DBMS_OUTPUT.PUT_LINE('The quantity for product ' || prod || ' is now '|| qt); -- shows a
message to confirm the qty has been updated
COMMIT;
END updqty;
```

Before picture of the PRODUCT table

EDIT	PART_NO	SUPPLIER	DESCRIPTION	SALE_PRICE	SUP_PRICE	QTY
	101	3026	Ericsson B and R Mgm	1000	50.5	65
	125	3025	Oracle Acme Packet 3820	650	35.75	60
	175	3030	Nokia Lumia 1020	559	15.85	29
	188	3021	Sony X25 Scanner	420	45	37
	190	3024	Red Hat Storage Server	800	65	20
	255	3028	Phillips X255 Printer	650	40.25	40
	295	3023	HP DL320 Server	1800	75	19
	320	3022	IBM 6182 Plotter	1250	80	28
	520	3029	Apple MacPro Dual GPU	3990	125	22
	521	3020	HP DL380p Gen8 Svr	2800	45.95	26
	860	3027	Samsung LT Cluster	2450	85.99	40

row(s) 1 - 11 of 11

Testing the procedure

```
BEGIN
    myproject.updqty(520,15);
END;
/
```

```
user@guestOS: ~

SQL> BEGIN   -- checks if there is an engineer for product 101 - engineer should
 be available
        myproject.checkengineer(101);
END;
/  2    3    4
Engineer 301 is available.

PL/SQL procedure successfully completed.

SQL> BEGIN
        myproject.updqty(520,15);
END;
/  2    3    4
The quantity for product 520 is now 37

PL/SQL procedure successfully completed.

SQL>
```

Checking the table after the procedure – part_no 520 has now quantity of 37

EDIT	PART_NO	SUPPLIER	DESCRIPTION	SALE_PRICE	SUP_PRICE	QTY
	101	3026	Ericsson B and R Mgm	1000	50.5	65
	125	3025	Oracle Acme Packet 3820	650	35.75	60
	175	3030	Nokia Lumia 1020	559	15.85	29
	188	3021	Sony X25 Scanner	420	45	37
	190	3024	Red Hat Storage Server	800	65	20
	255	3028	Phillips X255 Printer	650	40.25	40
	295	3023	HP DL320 Server	1800	75	19
	320	3022	IBM 6182 Plotter	1250	80	28
	520	3029	Apple MacPro Dual GPU	3990	125	37
	521	3020	HP DL380p Gen8 Svr	2800	45.95	26
	860	3027	Samsung LT Cluster	2450	85.99	40

row(s) 1 - 11 of 11

```
-- third procedure - procedure that activates a given quote into a support
contract and throws errors if the start date is in the past,
PROCEDURE actquote (ordID IN NUMBER)
IS
qtype NUMBER(1);   -- stores the document type
error NUMBER(1);   -- used to check the type of the error to give the
appropriate message
sdate DATE;   -- stores the start date
odate DATE;   -- stores the order date
now DATE;    -- stores the current date
quote_exc EXCEPTION;    -- throws the exception when error is encountered
BEGIN
SELECT type INTO qtype FROM ORDER_CALL WHERE orderID=ordID;  -- selects the
type of the given document
SELECT start_date INTO sdate FROM ORDER_CALL WHERE orderID=ordID;  --
assigns the start date
SELECT order_date INTO odate FROM ORDER_CALL WHERE orderID=ordID;   --
assigns the order date
SELECT sysdate INTO now FROM DUAL;    -- assigns the actual date into now
IF qtype=3 THEN           -- if type is 3 (active contract) it raises the
error
    error:=1;          -- assigns the error case
    RAISE quote_exc;
ELSE
    IF (now-sdate)>0 THEN        -- if the quote has the start date in past
it raises the error
        error:=2;
        RAISE quote_exc;
    ELSIF (now-odate)>30 THEN    --  if the quote has expired it raises the
error
        error:=3;
        RAISE quote_exc;
    ELSE                              -- it none of the above cases are met,
it updates the type into 3 - activates the quote into a support contract
        UPDATE ORDER_CALL SET
        type=3
        WHERE ORDER_CALL.orderID=ordID;
    END IF;
END IF;
EXCEPTION                -- the exception
WHEN quote_exc THEN
    IF error=1 THEN     -- checks the error case and outputs a message
accordingly
        DBMS_OUTPUT.PUT_LINE('Quote '|| ordID || ' is already active.');
    ELSIF error=2 THEN
        DBMS_OUTPUT.PUT_LINE('Quote '|| ordID || ' has a start date in the
past and cannot be activated. Please create a new quote with a future start
date.');
    ELSE
        DBMS_OUTPUT.PUT_LINE('Quote '|| ordID || ' has expired. Please
create a new quote.');
    END IF;
COMMIT;
END actquote;
```

```
BEGIN -- the quote is already active so it should return error 1
    myproject.actquote(100040);
END;
/
```

```
user@guestOS: ~
/  2    3    4
        actquote.updqty(100040);
                *
ERROR at line 2:
ORA-06550: line 2, column 11:
PLS-00225: subprogram or cursor 'ACTQUOTE' reference is out of scope
ORA-06550: line 2, column 2:
PL/SQL: Statement ignored


SQL> BEGIN -- the quote is already active so it should return error 1
        myproject.actquote(100040);
END;
/  2    3    4
Quote 100040 is already active.

PL/SQL procedure successfully completed.

SQL>
```

```
BEGIN -- the quote has a start date in past so it should raise exception and return error 2
  myproject.actquote(100078);
END;
/
```

```
user@guestOS: ~
SQL> BEGIN -- the quote is already active so it should return error 1
        myproject.actquote(100040);
END;
/  2    3    4
Quote 100040 is already active.

PL/SQL procedure successfully completed.

SQL> BEGIN -- the quote has a start date in past so it should raise exception an
d return error 2
        myproject.actquote(100078);
END;
/  2    3    4
Quote 100078 has a start date in the past and cannot be activated. Please create
a new quote with a future start date.

PL/SQL procedure successfully completed.

SQL>
```

```
BEGIN  -- the quote should be activated
   myproject.actquote(100145);
END;
/
```

```
user@guestOS: ~
SP2-0042: unknown command "/BEGIN" - rest of line ignored.
SP2-0044: For a list of known commands enter HELP
and to leave enter EXIT.
SQL> SP2-0734: unknown command beginning "myproject...." - rest of line ignored.
SQL> SP2-0042: unknown command "END" - rest of line ignored.
SQL> Quote 100082 has a start date in the past and cannot be activated. Please c
reate
a new quote with a future start date.

PL/SQL procedure successfully completed.

SQL> BEGIN  -- the quote should be activated
        myproject.actquote(100145);
END;
/  2    3    4

PL/SQL procedure successfully completed.

SQL>
```

	100145	04/19/2014	3	05/02/2014	02/22/2016	0	1530

```
-- fourth  procedure - this procedure actiates all the quotes created by a given ordering contact into active support contracts; it creates a savepoint at the beginning of the procedure, and starts activating the quotes; if a quote cannot be activated, it rolls back to the savepoint
PROCEDURE activateallcontracts (ordcontact IN NUMBER)
IS
now DATE;       -- stores the current date
nrofquotes NUMBER(3);    -- stores the number of quotes that have been activated into contracts
errors NUMBER(2);       -- stores the error type
CURSOR cur_oc IS SELECT * FROM ORDER_CALL WHERE contactID=ordcontact AND type=1;   -- cursor that checks all the quotes from order_call
order_row cur_oc%ROWTYPE;
BEGIN
SELECT sysdate INTO now FROM DUAL;   -- stores current date into now
SAVEPOINT SAVENOW;   -- creates a savepoint
OPEN cur_oc;
FETCH cur_oc INTO order_row;
nrofquotes:=0;      -- declares the nr of quotes to 0
errors := 0;
<<activatecontract>>
WHILE cur_oc%FOUND LOOP   -- loops the cursor through the table
   IF (now-order_row.start_date)>0 OR (now-order_row.order_date)>30 THEN  -- checks if the quote has expired or has a start date in the past
      DBMS_OUTPUT.PUT_LINE('Quote '|| order_row.orderID || ' could not been activated.');
      errors := errors+1;
   ELSE    -- if the quote can be activated, it activates it
      UPDATE ORDER_CALL SET ORDER_CALL.type=3 WHERE ORDER_CALL.orderID=order_row.orderID;
      DBMS_OUTPUT.PUT_LINE('Quote '|| order_row.orderID || ' has been activated.');
      nrofquotes := nrofquotes+1;    -- increases the nr of activated quotes
   END IF;
   FETCH cur_oc INTO order_row; -- read the next row

END LOOP;     -- ends the loop
CLOSE cur_oc;  -- closes the cursor after going through the whole table
IF errors>0 THEN    -- checks if there are any quotes that cannot be activated, outputs a message and rolls back to the savepoint
   DBMS_OUTPUT.PUT_LINE('The qoutes could not been activated. Please update the quotes and try to activate them after the update.');
   ROLLBACK TO SAVEPOINT SAVENOW;    -- rolls back to the savepoint
ELSE
   DBMS_OUTPUT.PUT_LINE(nrofquotes || ' quotes have been activated.');   -- if there were no errors, outputs the nr of quotes activated
END IF;
COMMIT;
END activateallcontracts;
```

```
BEGIN  -- quote 100143 has a start date in the past so it cannot be activated and should throw and error message
   myproject.activateallcontracts(1567);
END;
/
```

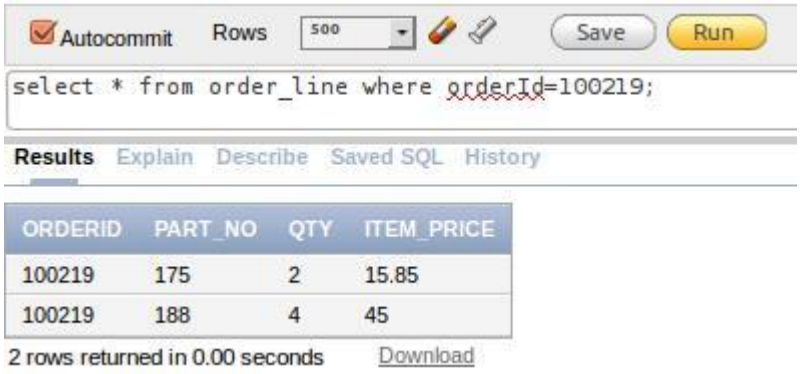
```
user@guestOS: ~

PL/SQL procedure successfully completed.

SQL> commit
  2  ;

Commit complete.

SQL> BEGIN
        myproject.activateallcontracts(1567);
END;
/  2    3    4
Quote 100143 could not been activated.
The qoutes could not been activated. Please update the quotes and try to
activate them after the update.

PL/SQL procedure successfully completed.

SQL>
```

```
-- there are 2 quotes for this customer that can be activated
BEGIN
    myproject.activateallcontracts(1500);
END;
/
```

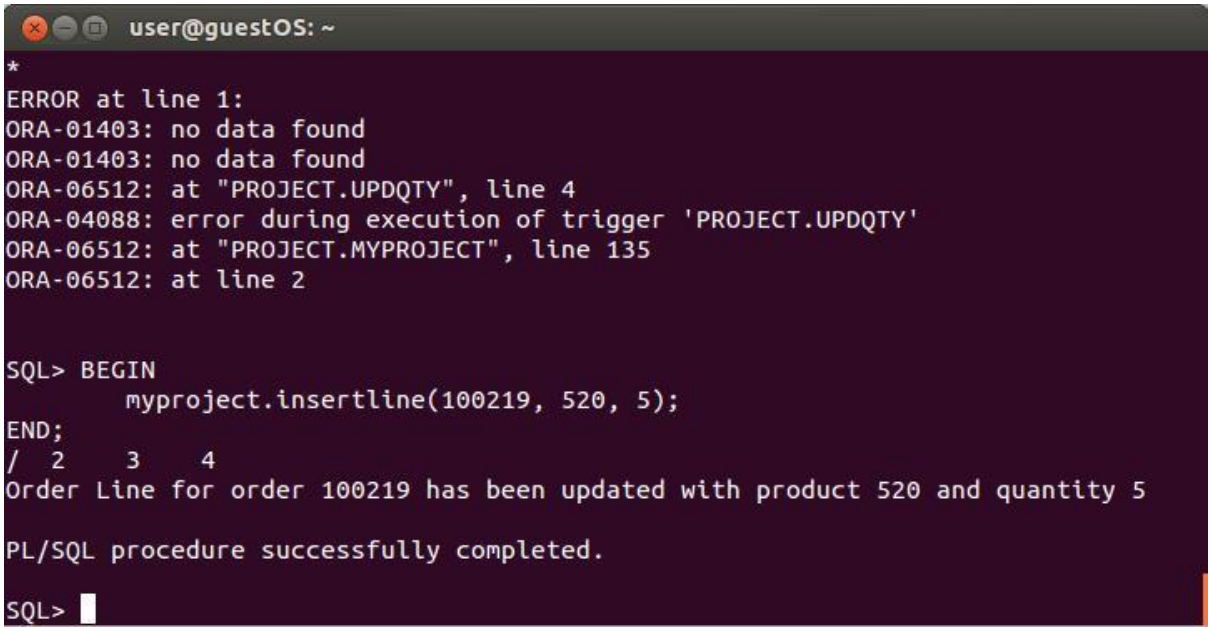
```
user@guestOS: ~
END;
/  2    3    4
Quote 100143 could not been activated.
The qoutes could not been activated. Please update the quotes and try to
activate them after the update.

PL/SQL procedure successfully completed.

SQL> BEGIN
        myproject.activateallcontracts(1500);
END;
/  2    3    4
Quote 100044 has been activated.
Quote 100047 has been activated.
2 quotes have been activated.

PL/SQL procedure successfully completed.

SQL>
```

```
-- fifth procedure - this procedure inserts a new order line into the
ORDER_LINE table based on the order ID, the part number and the quantity
given
PROCEDURE insertline (ordID IN NUMBER, prod IN NUMBER, qt IN NUMBER)
IS
BEGIN
INSERT INTO ORDER_LINE (orderID, part_no, qty) VALUES (ordID, prod, qt);
-- inserts the values into the ORDER_LINE table and outputs a message
DBMS_OUTPUT.PUT_LINE('Order Line for order '|| ordID || ' has been updated
with product '|| prod || ' and quantity ' || qt);
END insertline;
END;
/

BEGIN
   myproject.insertline(100219, 520, 5);
END;
/
```

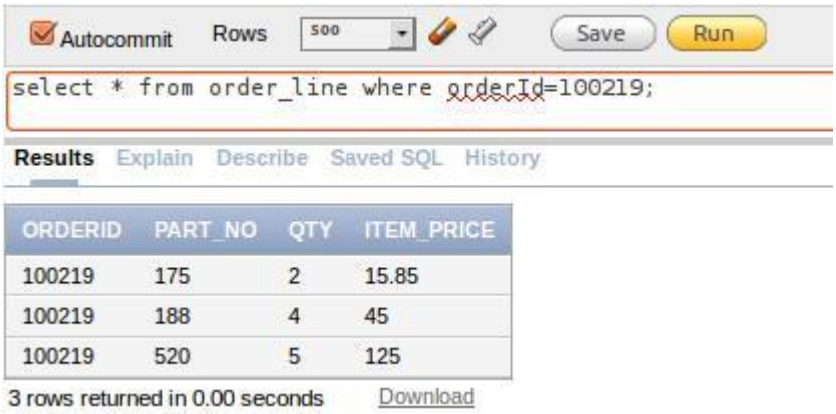
The entries in the order line table for order id 100219:



Procedure has been run:



The entries in the order line table for order id 100219 after running the procedure:

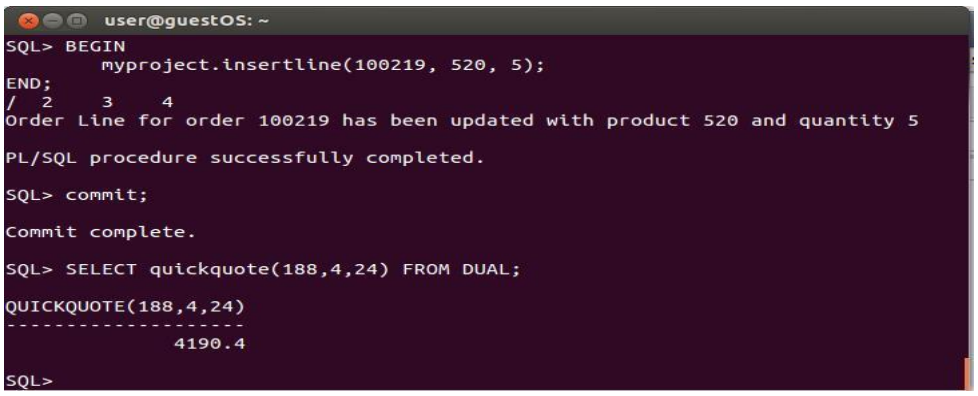


7. 2 PL/SQL functions

```
-- This function eases a sales person's job when a customer asks for a quick quote for only one product and returns the price; for example it can be used whenever a sales is communicating on the phone with a customer or whenever the sales person has a meeting with a customer
CREATE OR REPLACE FUNCTION quickquote (product IN NUMBER, qty IN NUMBER, len IN NUMBER)
RETURN NUMBER
IS
   discnt NUMBER(1);   -- stores the discount that has to be applied
   price NUMBER(5,2);   -- stores the price of that
BEGIN
   discnt :=0;    -- sets the discount to 0
   SELECT PRODUCT.sup_price
      INTO price
      FROM PRODUCT
      WHERE PRODUCT.part_no=product;
   IF (FLOOR(len/12)<2) THEN     -- checks if the length of the quote is less thant 2 years and no discount should be applied
      discnt:=0;      -- sets the discount to 0
   ELSE               -- if the quote has a length greater than 2 years checks which discount should be applied by the length and sets the discnt variable accordingly
      CASE
         WHEN FLOOR(len/12)=2 THEN discnt:=3;
         WHEN FLOOR(len/12)=3 THEN discnt:=4;
         WHEN FLOOR(len/12)=4 THEN discnt:=5;
         ELSE discnt:=7;
      END CASE;
   END IF;
   RETURN ((price*qty*len)-(price*qty*len*(discnt/100)));   -- returns the price of the quote
END quickquote;
/
```

Testing the function:

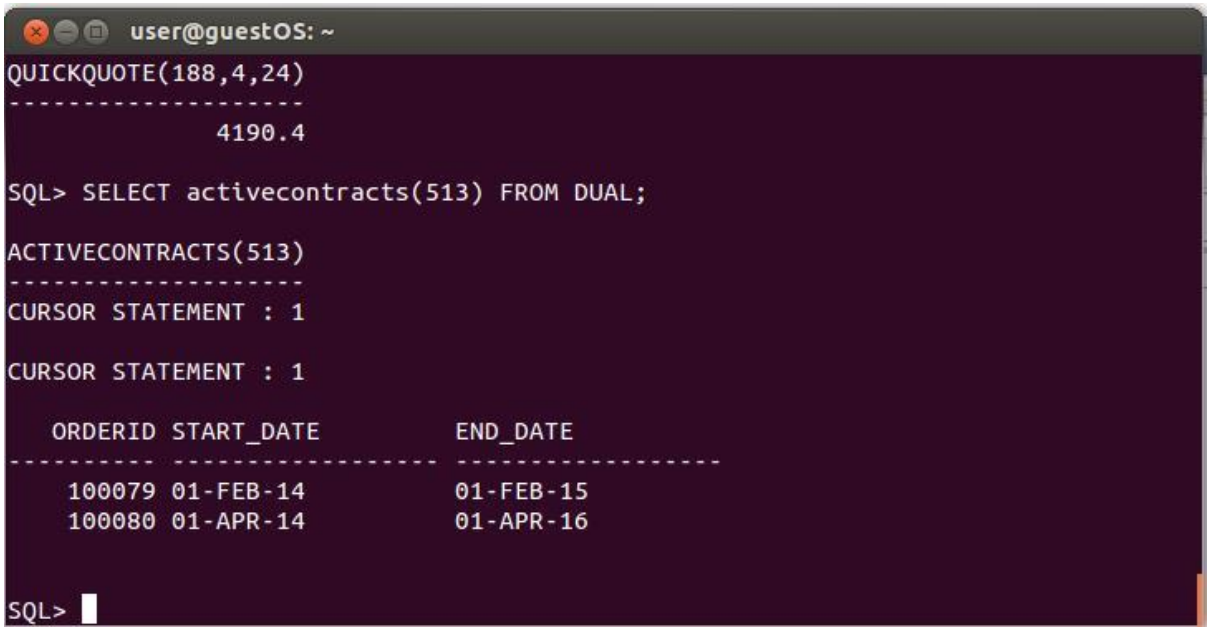
```
SELECT quickquote(188,4,24) FROM DUAL;
```



```
-- The function returns a table with all the active contracts of a customer
CREATE OR REPLACE FUNCTION activecontracts (customer IN NUMBER)    -- requires the customer ID
RETURN SYS_REFCURSOR          -- uses the sys_refcursor to return the recordset
AS
   contract SYS_REFCURSOR;   -- declares the sys_refcursror contrac
BEGIN
   OPEN contract          -- opens the cursor
   FOR SELECT ORDER_CALL.orderID, ORDER_CALL.start_date, ORDER_CALL.end_date --selects the columns required
      FROM ORDER_CALL
      JOIN CONTACT_PERSON
      ON ORDER_CALL.contactID=CONTACT_PERSON.contactID
      AND ORDER_CALL.type=3     -- selects only the entries with type of the order_call (3 means it is a support contract)
      AND sysdate>start_date    -- selects only the entries with the start date in the past
      AND sysdate<end_date     -- selects only the entries with end date in future    - this means the contract is active
      AND customer=CONTACT_PERSON.customerID;
   RETURN contract;     -- returns the recordsets
END activecontracts;
/
```

Testing the function:

```
SELECT activecontracts(513) FROM DUAL;
```

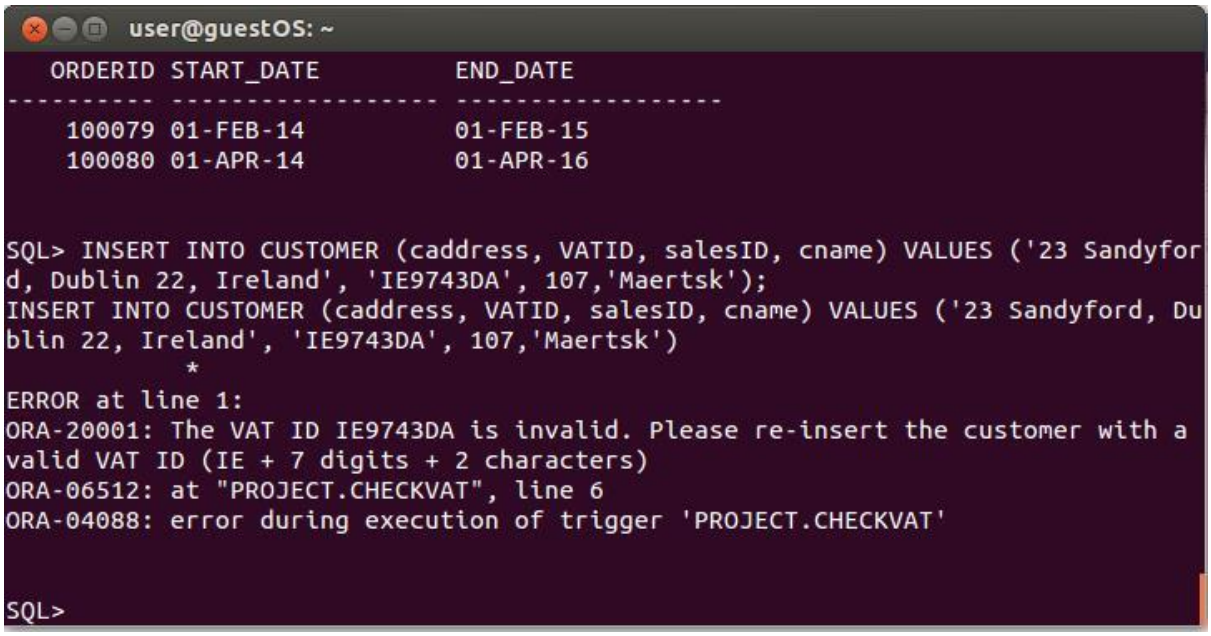


8. 3 Triggers (at least 1 before, and at least 1 after)

```
-- After Trigger that checks if the VAT ID inserted is valid; I assumed the standard VAT ID format for Ireland is IE + 7 digits and 2 letters (Ex: IE1234567FA).
CREATE OR REPLACE TRIGGER checkVAT AFTER    -- trigger declaration
INSERT OR UPDATE ON CUSTOMER
FOR EACH ROW
DECLARE
   vat NVARCHAR2(15) := :new.vatID;    -- stores the new value inserted for the VAT ID field in vat variable
BEGIN
   IF vat NOT LIKE 'IE[0-9]__' AND LENGTH(vat) <> 11 THEN  -- checks the vat for the VAT ID format and its length (has to be 11)
   RAISE_APPLICATION_ERROR(-20001, 'The VAT ID ' || :new.vatID || ' is invalid. Please re-insert the customer with a valid VAT ID (IE + 7 digits + 2 characters)');  -- if vat does not have the required format it raises an application error and the new row is not stored
   END IF;
END;
/
```

Testing trigger by insertion of a customer with a wrong VAT ID:

```
INSERT INTO CUSTOMER (caddress, VATID, salesID, cname) VALUES ('23 Sandyford, Dublin 22, Ireland', 'IE9743DA', 107,'Maertsk');
```



```
-- Before Trigger that checks the discount required for contracts and applies it
CREATE OR REPLACE TRIGGER mydisc BEFORE
INSERT OR UPDATE ON ORDER_CALL FOR EACH ROW WHEN (new.start_date is NOT NULL)
DECLARE
   dur NUMBER(2);   -- variable that stores the duration (length) in years
BEGIN
   dur:=FLOOR(MONTHS_BETWEEN(:new.end_date,:new.start_date)/12); -- calculates the length in
years
   IF dur < 2 THEN   -- if the length is less than 2 years it sets the new discount value to 0
     :new.discount := 0;
   ELSE          -- if the length is greater than 2 years sets the discount accordingly
     CASE
       WHEN dur=2 THEN :new.discount := 3;
       WHEN dur=3 THEN :new.discount := 4;
       WHEN dur=4 THEN :new.discount := 5;
       WHEN dur>=5 THEN :new.discount := 7;
     END CASE;
   END IF;
END;
/
```

Testing the trigger by inserting a new order with the length of 4 years in the order call table and checking if the discount is set accordingly

```
INSERT INTO ORDER_CALL (order_date, type, start_date, end_date, contactID)
VALUES ('26-APR-2012', 1, '01-JUN-2014', '01-JUN-2018', 1502);
```

```
SQL> INSERT INTO ORDER_CALL (order_date, type, start_date, end_date, contactID)
VALUES ('26-APR-2012', 1, '01-JUN-2014', '01-JUN-2018', 1502);

1 row created.

SQL>
```

EDIT	ORDERID	ORDER_DATE	TYPE	START_DATE	END_DATE	DISCOUNT	CONTACTID
	100210	02/25/2014	3	04/25/2014	05/22/2015	0	1534
	100211	03/28/2014	3	03/24/2014	11/20/2017	4	1536
	100212	03/06/2014	3	04/28/2014	05/17/2016	3	1560
	100213	03/09/2014	3	02/22/2014	07/06/2015	0	1504
	100214	02/23/2014	3	02/12/2014	02/12/2016	3	1511
	100215	03/01/2014	1	03/16/2014	06/17/2015	0	1561
	100216	03/22/2014	3	04/08/2014	09/05/2015	0	1510
	100217	04/09/2014	1	06/02/2014	01/25/2016	0	1507
	100218	03/10/2014	3	05/24/2014	11/22/2017	4	1526
	100219	02/21/2014	3	02/23/2014	03/10/2017	4	1527
	100220	04/26/2012	1	06/01/2014	06/01/2018	5	1502

row(s) 106 - 116 of 116

```
-- Before Trigger that updates the qty in the Product table based on the products ordered in the Order_Call table and sets the item_prices in the Order_Line table based on the type of the order and the respective sale prices or support price from the Product table
CREATE OR REPLACE TRIGGER updqty BEFORE   -- trigger declaration
INSERT OR UPDATE
ON ORDER_LINE
FOR EACH ROW
DECLARE
   ortype NUMBER(5);       -- stores the type of the order so that can modify the quantity only for type 2 (sale orders)
BEGIN
   SELECT type
      INTO ortype
      FROM ORDER_CALL
      WHERE order_call.orderID=:NEW.orderID;
   IF (ortype=2) THEN       -- checks if the order is of type 2
      UPDATE PRODUCT SET qty = qty - :NEW.qty WHERE part_no=:NEW.part_no;   -- updates the quantity in the product table by substracting the quantity of the newly added products
      SELECT sale_price  -- updates the item price in the order_line table based on the price of the product from the product table
         INTO :NEW.item_price
         FROM PRODUCT
         WHERE part_no=:NEW.part_no;
   ELSE        -- if it is not an order sale, it only updates the price of the newly added product and doesn't modify the quantity
      SELECT sup_price
         INTO :NEW.item_price
         FROM PRODUCT
         WHERE part_no=:NEW.part_no;
   END IF;
END;
/
```

Testing the trigger by inserting a new row in the Order_Line table, for a support order (the quantity doesn’t have to change):

```
INSERT INTO ORDER_LINE (orderID,part_no,qty) VALUES (100220,520,1);
```

Product table before the insert statement:

EDIT	PART_NO	SUPPLIER	DESCRIPTION	SALE_PRICE	SUP_PRICE	QTY
	101	3026	Ericsson B and R Mgm	1000	50.5	65
	125	3025	Oracle Acme Packet 3820	650	35.75	60
	175	3030	Nokia Lumia 1020	559	15.85	29
	188	3021	Sony X25 Scanner	420	45	37
	190	3024	Red Hat Storage Server	800	65	20
	255	3028	Phillips X255 Printer	650	40.25	40
	295	3023	HP DL320 Server	1800	75	19
	320	3022	IBM 6182 Plotter	1250	80	28
	520	3029	Apple MacPro Dual GPU	3990	125	37
	521	3020	HP DL380p Gen8 Svr	2800	45.95	26
	860	3027	Samsung LT Cluster	2450	85.99	40

row(s) 1 - 11 of 11

Insert statement:

```
SQL> INSERT INTO ORDER_LINE (orderID,part_no,qty) VALUES (100220,520,1);

1 row created.
```

Product table after the insert statement:

EDIT	PART_NO	SUPPLIER	DESCRIPTION	SALE_PRICE	SUP_PRICE	QTY
	101	3026	Ericsson B and R Mgm	1000	50.5	65
	125	3025	Oracle Acme Packet 3820	650	35.75	60
	175	3030	Nokia Lumia 1020	559	15.85	29
	188	3021	Sony X25 Scanner	420	45	37
	190	3024	Red Hat Storage Server	800	65	20
	255	3028	Phillips X255 Printer	650	40.25	40
	295	3023	HP DL320 Server	1800	75	19
	320	3022	IBM 6182 Plotter	1250	80	28
	520	3029	Apple MacPro Dual GPU	3990	125	37
	521	3020	HP DL380p Gen8 Svr	2800	45.95	26
	860	3027	Samsung LT Cluster	2450	85.99	40

row(s) 1 - 11 of 11

Order Line table:

EDIT	ORDERID	PART_NO	QTY	ITEM_PRICE
	100220	520	1	125

row(s) 196 - 196 of 196

Testing the trigger for a sale order:

INSERT INTO ORDER_LINE **(**orderID,part_no,qty**) VALUES (**100188,520,3**);**

```
SQL> INSERT INTO ORDER_LINE (orderID,part_no,qty) VALUES (100188,520,3);

1 row created.

SQL>
```

Checking the order line table – the price has been updated:

EDIT	ORDERID	PART_NO	QTY	ITEM_PRICE
	100220	520	1	125
	100188	520	3	3990

row(s) 196 - 197 of 197

Checking the product table – the quantity has been updated:

EDIT	PART_NO	SUPPLIER	DESCRIPTION	SALE_PRICE	SUP_PRICE	QTY
	101	3026	Ericsson B and R Mgm	1000	50.5	65
	125	3025	Oracle Acme Packet 3820	650	35.75	60
	175	3030	Nokia Lumia 1020	559	15.85	29
	188	3021	Sony X25 Scanner	420	45	37
	190	3024	Red Hat Storage Server	800	65	20
	255	3028	Phillips X255 Printer	650	40.25	40
	295	3023	HP DL320 Server	1800	75	19
	320	3022	IBM 6182 Plotter	1250	80	28
	520	3029	Apple MacPro Dual GPU	3990	125	34
	521	3020	HP DL380p Gen8 Svr	2800	45.95	26
	860	3027	Samsung LT Cluster	2450	85.99	40

row(s) 1 - 11 of 11